Submitted to Foundations and Trends in Databases

Machine Knowledge: Creation and Curation of Comprehensive Knowledge Bases

Gerhard Weikum
Max Planck Institute for Informatics
weikum@mpi-inf.mpg.de

Luna Dong
Amazon
lunadong@amazon.com

Simon Razniewski
Max Planck Institute for Informatics
srazniew@mpi-inf.mpg.de

Fabian Suchanek
Telecom Paris University
suchanek@telecom-paris.fr

September 25, 2020

Abstract

Equipping machines with comprehensive knowledge of the world's entities and their relationships has been a long-standing goal of AI. Over the last decade, large-scale knowledge bases, also known as knowledge graphs, have been automatically constructed from web contents and text sources, and have become a key asset for search engines. This machine knowledge can be harnessed to semantically interpret textual phrases in news, social media and web tables, and contributes to question answering, natural language processing and data analytics.

This article surveys fundamental concepts and practical methods for creating and curating large knowledge bases. It covers models and methods for discovering and canonicalizing entities and their semantic types and organizing them into clean taxonomies. On top of this, the article discusses the automatic extraction of entity-centric properties. To support the long-term life-cycle and the quality assurance of machine knowledge, the article presents methods for constructing open schemas and for knowledge curation. Case studies on academic projects and industrial knowledge graphs complement the survey of concepts and methods.

Contents

1 What Is This All About

1.1 Motivation

Enhancing computers with "machine knowledge" that can power intelligent applications is a long-standing goal of computer science [323]. This formerly elusive vision has become practically viable today, made possible by major advances in *knowledge harvesting*. This comprises methods for turning noisy Internet content into crisp knowledge structures on entities and relations. The knowledge harvesting methodology has enabled the automatic construction of *knowledge bases (KB)*: collections of machine-readable facts about the real world. Today, publicly available KBs provide millions of entities (such as people, organizations, locations and creative works like books, music etc.) and billions of statements about them (such as who studied where, which country has which capital, or which singer performed which song). Proprietary KBs deployed at major companies comprise knowledge at an even larger scale, with one or two orders of magnitude more entities.

A prominent use case where knowledge bases have become a key asset is web search. When we send a query like "dylan protest songs" to Baidu, Bing or Google, we obtain a crisp list of songs such as Blowin' in the Wind, Masters of War, or A Hard Rain's a-Gonna Fall. So the search engine automatically detects that we are interested in facts about an individual entity – Bob Dylan in this case – and ask for specifically related entities of a certain type – protest songs – as answers. This is feasible because the search engine has a huge knowledge base in its back-end data centers, aiding in the discovery of entities in user requests (and their contexts) and in finding concise answers.

The KBs in this setting are centered on individual entities, containing (at least) the following backbone information:

- entities like people, places, organizations, products, events, such as
  `Bob Dylan`, the `Stockholm City Hall`, the `2016 Nobel Prize Award Ceremony`
- the semantic classes to which entities belong, for example
  ⟨`Bob Dylan, type, singer-songwriter`⟩,
  ⟨`Bob Dylan, type, poet`⟩
- relationships between entities, such as
  ⟨`Bob Dylan, created, Blowin' in the Wind`⟩,
  ⟨`Bob Dylan, won, Nobel Prize in Literature`⟩

Some KBs also contain validity times such as

- ⟨`Bob Dylan, married to, Sara Lownds, [1965,1977]`⟩

This temporal scoping is optional, but very important for the life-cycle management of a KB as the real world evolves over time. In the same vein of long-term quality assurance, KBs may also contain constraints and provenance information.

1.2 History of Large-Scale Knowledge Bases

The concept of a comprehensive KB goes back to pioneering work in Artificial Intelligence on universal knowledge bases in the 1980s and 1990s, most notably, the *Cyc* project at MCC in Austin [322] and the *WordNet* project at Princeton [159]. However, these knowledge collections have been hand-crafted and curated manually. Thus, the knowledge acquisition was inherently limited in scope and scale. With the Semantic Web vision in the early 2000s, domain-specific *ontologies* [551] have been developed, but these were also manually created. In the first decade of the 2000s, *automatic knowledge harvesting* from Web and text sources became a major research avenue, and has made substantial practical impact. Knowledge harvesting is the core methodology for the automatic construction of large knowledge bases, going beyond manually compiled knowledge collections like Cyc or WordNet.

These achievements are rooted in academic research and community projects. Salient projects that started in the 2000s are DBpedia [18], Freebase [51], KnowItAll [153], WebOfConcepts [110], WikiTaxonomy [456] and YAGO [562]. More recent projects with publicly available data include BabelNet [418], ConceptNet [548], DeepDive [527], EntityCube (aka. Renlifang) [425], KnowledgeVault [129], NELL [71], Probase [631], WebIsALOD [232], Wikidata [600], and XLore [617]. More on the history of KB technology can be found in the overview article [245].

At the time of writing this survey, the largest general-purpose KBs with publicly accessible contents are Wikidata (wikidata.org), BabelNet (babelnet.org), DBpedia (dbpedia.org), and YAGO (yago-knowledge.org). They contain millions of entities, organized in hundreds to hundred thousands of semantic classes, and hundred millions to billions of relational statements on entities. These and other knowledge resources are interlinked at the entity level, forming the Web of Linked Open Data [225, 244].

Over the 2010s, knowledge harvesting has been adopted at big industrial stakeholders [429], and large KBs have become a key asset in a variety of commercial applications, including semantic search (see, e.g., [35, 484]), analytics (e.g., aggregating by entities), recommendations (see, e.g., [195]), and data integration (i.e., to combine heterogeneous datasets in and across enterprises). Examples are the Google Knowledge Graph [536], the use of KBs in IBM Watson [163], the Amazon Product Graph [127, 133], the Alibaba e-Commerce Graph [360], the Baidu Knowledge Graph [26], Microsoft Satori [467], Wolfram Alpha [248] as well as domain-specific knowledge bases in business, finance, life sciences, and more (e.g., at Bloomberg [375]).

In addition, KBs have found wide use as a source of distant supervision for a variety of tasks in natural language processing, such as entity linking.

1.3 Application Use Cases

Knowledge bases enable or enhance a wide variety of applications.

Semantic Search and Question Answering:

All major search engines have some form of KB as a background asset. Whenever a user's information need centers around an entity or a specific type of entities, such as singers, songs, tourist locations, companies, products, sports events etc., the KB can return a precise and concise list of entities rather than merely giving "ten blue links" to web pages. The earlier example of asking for "dylan protest songs" is typical for this line of semantic search. Even when the query is too complex or the KB is not complete enough to enable entity answers, the KB information can help to improve the ranking of web-page results by considering the types and other properties of entities. Similar use cases arise in enterprises as well, for example, when searching for customers or products with specific properties, or when forming a new team with employees who have specific expertise and experience.

An additional step towards user-friendly interfaces is question answering (QA) where the user poses a full-fledged question in natural language and the system aims to return crisp entity-style answers from the KB or from a text corpus or a combination of both. An example for KB-based QA is "Which songs written by Bob Dylan received Grammys?"; answers include All Along the Watchtower, performed by Jimi Hendrix, which received a Hall of Fame Grammy Award. An ambitious example that probably requires tapping into both KB and text would be "Who filled in for Bob Dylan at the Nobel Prize ceremony in Stockholm?"; the answer is Patti Smith.

Overviews on semantic search and question answering with KBs include [35, 119, 320, 484, 589].

Language Understanding and Text Analytics:

Both written and spoken language are full of ambiguities. Knowledge is the key to mapping surface phrases to their proper meanings, so that machines interpret language as fluently as humans. AI-style use cases include machine translation, and conversational assistants like chatbots. Prominent examples include Amazon's Alexa, Apple's Siri, and Google's Assistant and new chatbot initiatives [2], and Microsoft's Cortana.

In these applications, world knowledge plays a crucial role. Consider, for example, sentences like "Jordan holds the record of 30 points per match" or "The forecast for Jordan is a record high of 110 degrees". The meaning of the word "Jordan" can be inferred by having world knowledge about the basketball champion Michael Jordan and the middle-east country Jordan.

Understanding entities (and their properties and relations) in text is also key to large-scale analytics over news articles, scientific publications, review forums, or social media discussions. For example, we can identify mentions of products (and associated consumer opinions), link them to a KB, and then perform comparative and aggregated studies. We can even incorporate filters and groupings on product categories, geographic regions etc., by combining the textual information with structured data from the KB or from product and customer databases. All this can be enabled by the KB as a clean and comprehensive repository of entities (see, e.g., [523] for a survey on the core task of entity linking).

A trending example of semantic text analytics is detecting gender bias in news and other online content (see, e.g., [565]). By identifying people, and the KB knowing their gender, we can compute statistics over male vs. female people in political offices or on company boards. If we also extract earnings from movies and ask the KB to give us actors and actresses, we can shed light into potential unfairness in the movie industry.

Visual Understanding:

For detecting objects and concepts in images (and videos), computer vision has made great advances using machine learning. The training data for these tasks are collections of semantically annotated images, which can be viewed as visual knowledge bases. The most well-known example is ImageNet ([115]) which has populated a subset of WordNet concepts with a large number of example images. A more recent and more advanced endeavor along these lines is VisualGenome ([295]). By themselves, these assets already go a long way, but their value can be further boosted by combining them with additional world knowledge.

For example, knowing that lions and tigers are both predators from the big cat family and usually prey on deer or antelopes, can help to automatically label scenes as "cats attack prey". Likewise, recognizing landmark sites such as the Brandenburg gate and having background knowledge about them (e.g., other sites of interest in their vicinity) helps to understand details and implications of an image. In such computer vision and further AI applications, a KB often serves as an informed prior for machine learning models, or as a reference for consistency (or plausibility) checks.

Data Cleaning:

Coping with incomplete and erroneous records in large heterogeneous data is a classical topic in database research (see, e.g., [472]). The problem is more timely and pressing than ever. Data scientists and business analysts want to rapidly tap into diverse datasets, for comparison, aggregation and joint analysis. So different kinds of data need to be combined and fused, more or less on the fly and thus largely depending on automated tools. This trend amplifies the crucial role of identifying and repairing missing and incorrect values.

In many cases, the key to spotting and repairing errors or to infer missing values is consistency across a set of records. For example, suppose that a database about music has a new tuple stating that Jeff Bezos won the Grammy Award. A background knowledge base would tell that Bezos is an instance of types like businesspeople, billionaires, company founders etc., but there is no type related to music. As the Grammy is given only for songs, albums and musicians, the tuple about Bezos is likely a data-entry error. In fact, the requirement that Grammy winners, if they are of type person, have to be musicians, can be encoded into a logical consistency constraint. Several KBs contain such consistency constraints. They typically include:

- type constraints, e.g.: a Grammy winner who belongs to the type `person` must also be an instance of type `musician` (or a sub-type),
- functional dependencies, e.g.: for each year and each award category, there is exactly one Grammy winner,
- inclusion dependencies, e.g.: composers are also musicians and thus can win a Grammy, and all Grammy winners must have at least one song to which they contributed (i.e., the set of Grammy winners is a subset of the set of people with at least one song),
- disjointness constraints, e.g.: songs and albums are disjoint, so no piece of music can simultaneously win both of these award categories for the Grammy,
- temporal constraints, e.g.: the Grammy award is given only to living people, so that someone who died in a certain year cannot win it in any later year.

Data cleaning as a key stage in data integration is surveyed by [256, 255] and the articles in [328].

1.4 Frequently Asked Questions

Are knowledge bases part of the Semantic Web, and can they be used only for Semantic Web applications?

The big revival of knowledge bases in this millenium originated from research projects in the Semantic Web community. This is why some design choices favor models and methods from the Semantic Web. For example, the RDF data model is popular among KBs, and the query language of choice is often close to the Sparql language rather than SQL. However, it is easy to move KBs into the ecosystem of other data models and their tool suites. In particular, KBs can be stored, accessed and managed by relational database systems as well, and can be used also with NoSQL platforms such as Apache Spark and with cloud-based services. Likewise, it is easy to combine KBs with popular tools for machine learning such as TensorFlow, SciPy etc.

How is knowledge harvesting related to the field of information extraction?

Information extraction (IE) (see, e.g., [511, 162, 208]) comprises methodologies for recognizing and semantically annotating meaningful units in natural language and other kinds of noisy contents (e.g., ad-hoc tables in web pages, or query-and-click logs), building on text mining and machine learning. Given an arbitrary input, IE aims at a best-effort job on computing value-added mark-up. Knowledge harvesting leverages IE methods, but it is output-driven. To construct a high-quality KB, judicious choices about input sources and extraction strategies are crucial considerations. For example, we often want to "pick low-hanging fruit" first, for high quality, and tap into noisier sources only subsequently and with specific focus and customized techniques.

Why do we need machine knowledge, when we already have end-to-end machine learning working so well?

Machine learning (ML), especially deep neural networks, work well when there is sufficient training data with gold-standard labels. However, there are fundamental reasons why ML alone is not a full solution. First, training data is the typical bottleneck when tackling new applications, so that a lot of time and money needs to be spent on compiling and organizing the relevant data. This cost arises in each and every application again and again. Machine knowledge is an easily re-usable, versatile asset that can simplify and accelerate these expensive steps. Second, even the best deep learning methods are far from near-human quality in identifying crisp statements in complex, noisy and ambiguous texts and other input sources. There are fundamental issues for these limitations: ML is based on the paradigm of learning with iid samples from a data distribution and applying the trained model to new samples from the same distribution (iid = independently identically distributed). In contrast, humans do not solely rely on situative data alone, but interpret observations and learn with a rich body of background experience – the human's general world knowledge. Therefore, machine learning and machine knowledge are complementary pillars of modern AI. The more a machine knows, the better it can learn; and better learning enables acquiring more and deeper knowledge.

Are knowledge bases simply some sort of databases?

Knowledge bases are data, and can, of course, be stored in standard databases. But their scope and use cases make them special in several ways. First, a KB is a reference repository of entities, types and vocabulary with open-ended scope, modeling broad domains or enterprise-wide knowledge or even aiming for universal encyclopedic coverage. To this end, KBs include a rich taxonomy of types, way more expressive than the usual databases. Second, as a consequence, KBs operate under the Open World Assumption, which allows data to be incomplete, and KBs are grown with continuous curation. Third, to fulfill these requirements, KBs need to continously adapt and enhance their schemas for types and properties, following the paradigm of agile *data spaces* [206] rather than traditional "schema first" databases. This aspect led to the pragmatic choice for the RDF data model with binary relations, to allow new types and new properties to be added in a light-weight manner.

Are knowledge bases useful for enterprises?

KBs are useful as reference data in many ways. They contain encyclopedic knowledge about the world's notable entities, including people, places, events, and organizations. This can serve as background knowledge in enterprises and their applications. In the travel and tourism industry, for example, KBs can contribute knowledge about vacation sites, natural or cultural points of interest, and geography. Several of the publicly available KBs are rich on this kind of geo-spatial knowledge. Industrial applications can combine this with their data about hotels, flights, commercial tours etc.

There are also KBs specifically geared for a vertical domain or a company. In the health domain, for example, KBs can collect background knowledge about diseases, drugs, symptoms, therapies and their properties. In a company, a KB can provide relevant knowledge about customers, products, product categories, sales regions, and so on.

Industrial applications require involving domain experts, within the enterprise. The methodology presented in this article will benefit such teams in companies on automating and accelerating their endeavors.

Can high-tech startups benefit from knowledge bases?

KBs contribute to methods and tools for language understanding, data cleaning, machine-learning-based AI, semantic reasoning, and knowledge modeling. Therefore, several startups have taken to offer KB technology as a product. Some of these turned into big success stories: Freebase (by Metaweb Technologies, Inc.) was acquired by Google and kick-started the Google Knowledge Graph, and DeepDive (by Lattice Data, Inc.) was acquired by Apple.

1.5 Outline

This article covers methods for automatically constructing and curating large knowledge bases from web and text sources. We hope that it will be useful for doctoral students and faculty interested in a wide spectrum of topics – from machine knowledge and data quality to machine learning and data science as well as applications in web content mining and natural language understanding. In addition, this article aims to be useful also for industrial researchers and practitioners working on semantic technologies for web, social media, or enterprise contents, including all kinds of applications where sense-making from text or semi-structured data is an issue. Prior knowledge on natural language processing or statistical learning is not required; we will introduce relevant methods as they are needed (or at least give specific pointers to literature).

The article is organized into ten chapters. Chapter 2 gives foundational basics on knowledge representation and discusses the design space for building a KB. Chapters 3, 4 and 5 cover the methodology for constructing the core of a KB that comprises entities and types. Chapter 3 discusses tapping premium sources with rich and clean semi-structured contents, and Chapter 4 addresses knowledge harvesting from textual contents. Chapter 5 specifically focuses on the important issue of canonicalizing entities into unique representations. Chapters 6 and 7 extend the scope of the KB by methods for discovering and extracting attributes of entities and relations between entities. Chapter 6 focuses on the case where a schema is designed upfront for the properties of interest. Chapter 7 discusses the case of discovering new property types for attributes and relations that are not (yet) specified in the KB schema. Chapter 8 discusses the issue of quality assurance for KB curation and the long-term maintenance of KBs. Chapter 9 presents several case studies on specific KBs including industrial knowledge graphs (KGs). We conclude in Chapter 10 with key lessons and an outlook on where the theme of machine knowledge may be heading.

2 Foundations and Architecture

2.1 Knowledge Representation

Knowledge bases, KBs for short, comprise salient information about entities, semantic classes to which entities belong, attributes of entities, and relationships between entities. When the focus is on classes and their logical connections such as subsumption and disjointness, knowledge repositories are often referred to as **ontologies**. In database terminology, this is referred to as the **schema**. The class hierarchy alone is often called a **taxonomy**. The notion of KBs in this article covers all these aspects of knowledge, including ontologies and taxonomies.

This chapter presents foundations for casting knowledge into formal representations. Knowledge representation has a long history, spanning decades of AI research, from the classical model of frames to recent variants of description logics. Overviews on this spectrum are given by [504] and [551]. In this article, we restrict ourselves to the knowledge representation that has emerged as a pragmatic consensus for entity-centric knowledge bases (see [564] for an extended discussion). More on the wide spectrum of knowledge modeling can be found in the survey [245].

2.1.1 Entities

The most basic element of a KB is an *entity*.

> An **entity** is any abstract or concrete object of fiction or reality.

This definition includes people, places, products and also events and creative works (books, poems, songs etc.), real people (living or dead) as well as fictional people (e.g., Harry Potter), and also general concepts such as empathy and Buddhism. KBs take a pragmatic approach: they model only entities that match their scope and purpose. A KB on writers and their biographies would include Shakespeare and his drama Macbeth, but it may not include the characters of the drama such as King Duncan or Lady Macbeth. However, a KB for literature scholars – who want to analyze character relationships in literature content – should include all characters from Shakespeare's works.

Individual Entities (aka. Named Entities): We often narrow down the set of entities of interest by emphasizing uniquely identifiable entities and distinguishing them from general concepts.

> An **individual entity** is an entity that can be uniquely identified against all other entities.

To uniquely identify a location, we can use its geo-coordinates: longitude and latitude with a sufficiently precise resolution. To identify a person, we would – in the extreme case – have to use the person's DNA sequence, but for all realistic purpose a combination of full name, birthplace and birthdate are sufficient. In practice, we are typically even coarser and just use location or person names when there is near-universal social consensus about what or who is denoted by the name. Wikipedia article names usually follow this principle of using names that are sufficiently unique. For these arguments, individual entities are also referred to as **named entities**.

Identifiers and Labels: To denote an entity unambiguously, we need a name that can refer to only a single entity. Such an *identifier* can be a unique name, but can also be specifically introduced keys such as URLs for web sites, ISBNs for books (which can even distinguish different editions of the same book), DOIs for publications, Google Scholar URLs or ORCID IDs for authors, etc. In the data model of the Semantic Web, **RDF** (for **Resource Description Framework**) [602], identifiers always take the form of URIs (Unique Resource Identifiers, a generalization of URLs).

> An **identifier** for an entity is a string of characters that uniquely denotes the entity.

As identifiers are not necessarily of a form that is nicely readable and directly interpretable by a human, we often want to have human-readable **labels** or **names** in addition. For example, the protagonist in Shakespeare's Hamlet, Prince Hamlet of Denmark, may have an unwieldy identifier like https://www.wikidata.org/wiki/Q2447542 (in the Wikidata KB), but we want to refer to him also by labels like "Prince Hamlet", "Prince Hamlet of Denmark" or simply "Hamlet". Labels may be ambiguous, however: "Hamlet" alone could also refer to his father, the King of Denmark, to the play itself (titled "Hamlet"), or to the Hamlet castle (which exists in reality). When an entity has several labels, they are called **synonyms** or **alias names** (different names, same meaning). When different entities share a label such as "Hamlet", this label is a **homonym** (same name, different meanings).

2.1.2 Classes (aka. Types)

Entities of interest often come in groups where all elements have a shared characteristic. For example, `Bob Dylan`, `Elvis Presley` and `Lisa Gerrard` are all musicians and singers. We capture this knowledge by organizing entities into **classes** or, synonomously, **types**.

> A **class** (or *type*) is a named set of entities that share a common trait. An element of that set is called an **instance** of the class.

For Dylan, Presley and Gerrard, classes of interest include: `musicians` and `singers` with all three as instances, `guitarists` with Dylan and Presley as instances, `men` with these two, `women` containing only Gerrard, and so on.

Note that an entity can belong to multiple classes, and classes can relate to each other in terms of their members as being disjoint (e.g. `men` and `women`), overlapping, or one subsuming the other (e.g., `musicians` and `guitarists`). Classes can be quite specific; for example `left-handed electric guitar players` could be a class in the KB containing Jimi Hendrix, Paul McCartney and others.

It is not always obvious whether something should be modeled as an entity or as a class. We could construct, for every entity, a singleton class that contains just this entity. Classes of interest typically have multiple instances, though. By this token, we do not consider general concepts such as love, Buddhism or pancreatic cancer as classes, unless we were interested in specific instances (e.g., the individual cancer of one particular patient).

Taxonomies (aka. Class Hierarchies):

By relating the instance sets of two classes, we can specify invariants that must hold between the classes, most notably, subsumption, also known as the **subclass/superclass** relation. By combining these pairwise invariants across all classes, we can thus construct a **class hierarchy**. We refer to this aspect of the KB as a **taxonomy**.

> Class A is a **subclass** of (is subsumed by) class B if all instances of A must also be instances of B.

For example, the classes `singers` and `guitarists` are subclasses of `musicians` because every singer and every guitarist is a musician. We say that class X is a *direct* subclass of Y if there is no other class that subsumes X and is subsumed by Y. Classes can have multiple superclasses, but there should not be any cycles in the subsumption relation. For example, `left-handed electric guitar players` are a subclass of both `left-handed people` and `guitarists`.

> A **taxonomy** is a directed acyclic graph, where the nodes are classes and there is an edge from class X to class Y if X is a direct subclass of Y.

Note that these invariants do not just *describe* the current instances of such class pairs, but actually *prescribe* that the invariant holds for all possible instance sets. So the taxonomy acts like a database *schema*, and is instrumental for keeping the KB consistent. In database terminology, a subclass/superclass pair is also called an **inclusion dependency**. In the Semantic Web, the **RDFS** extension of the RDF model (RDFS = RDF Schema) allows specifying such constraints [604]. Other Semantic Web models, notably, OWL, support also disjointness constraints (aka. mutual exclusion), for example, specifying that `men` and `women` are disjoint.

One of the largest taxonomic repositories is the **WordNet** lexicon [159], comprising more than 100,000 classes. Figure 2.1 visualizes an excerpt of the WordNet taxonomy. The nodes are classes, called *word senses* in WordNet, the edges indicate subsumption.

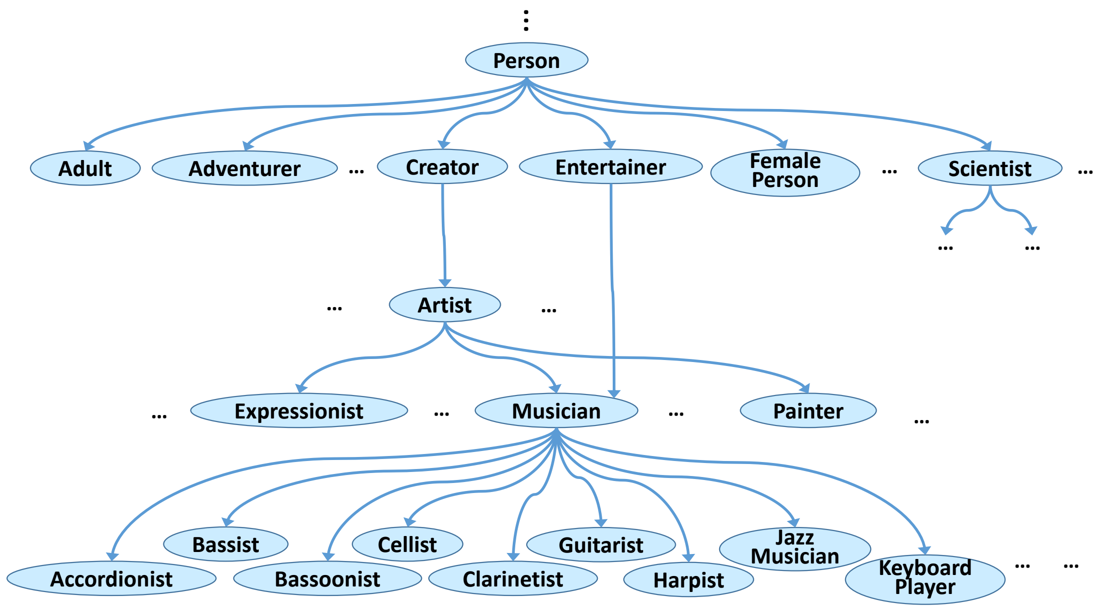


Figure 2.1: Excerpt from the WordNet Taxonomy

In linguistic terminology, this lexical relation is called **hypernymy**: an edge connects a more special class, called *hyponym*, with a generalized class, called *hypernym*. [203] gives an overview of this kind of lexical resources. Further examples of (potentially unclean) taxonomies include the Wikipedia category system or product catalogs.

Subsumption vs. Part-Whole Relation:
Class subsumption should not be confused or conflated with the relationship between parts and wholes. For example, a `soprano saxophone` is *part of* a `jazz band`. This does not mean, however, that every soprano saxophone *is a* jazz band. Likewise, New York is part of the USA, but `New York` is not a subclass of the `USA`.

Instance-of vs. Subclass-of:
Some KBs do not make a clear distinction between classes and instances, and they collapse the *instance-of* and *subclass-of* relations into a single **is-a hierarchy**. Instead of stating that `Bob Dylan` is an instance of the class `singers` and that `singers` are a subclass of `musicians`, they would view all three as general entities and connect them in a generalization graph.

2.1.3 Properties of Entities: Attributes and Relations

Entities have **properties** such as birthdate, birthplace and height of a person, prizes won, books, songs or software written, and so on. KBs capture these in the form of mathematical relations:

> A **relation** or **relationship** for the instances of classes $C_1, ..., C_n$ is a subset of the Cartesian product $C_1 \times ... \times C_n$, along with an identifier (i.e., unique name) for the relation.

For example, we can state the birthdate and birthplace of Bob Dylan in the relational form:

$\langle$`Bob Dylan`, `24-5-1941`, `Duluth (Minnesota)`$\rangle \in$ `birth`

where `birth` is the identifier of the relation. This instance of the `birth` relation is a ternary tuple, that is, it has three arguments: the person entity, the birthdate, and the birthplace. The underlying Cartesian product of the relation is `persons` $\times$ `dates` $\times$ `cities`.

In logical notation, we also write $R(x_1, ..., x_n)$ instead of $\langle x_1, ..., x_n\rangle \in R$, and we refer to R as a **predicate**. The number of arguments, n, is called the *arity* of R. The domain $C_1 \times ... \times C_n$ is also called the relation's **type signature**

As most KBs are of encyclopedic nature, the instances of a relation are often referred to as *facts*. We do not want to exclude knowledge that is not fact-centric (e.g., commonsense knowledge with a socio-cultural dimension); so we call relational instances more generally **statements**. The literature also speaks of **facts**, and sometimes uses the terminology *assertion* as well. For this article, the three terms *statement*, *fact* and *assertion* are more or less interchangeable.

In logical terms, statements are grounded expressions of first-order predicate logic (where "grounded" means that the expression has no variables). In the KB literature, the term "relation" is sometimes used to denote both the relation identifier R and an instance $\langle x_1, ..., x_n\rangle$. We avoid this ambiguity, and more precisely speak of the relation and its (relational) tuples.

Attributes of Entities:
In the above example about the `birth` relation, we made use of the class `dates`. By stating this, we consider individual dates, such as `24-5-1941`, entities. It is a design choice whether we regard numerical expressions like dates, heights or monetary amounts as entities or not. Often, we want to treat them simply as **values** for which we do not have any additional properties. In the RDF data model, such values are called **literals**. Strings such as nicknames of people (e.g., "Air Jordan") are another frequent type of literals.

We introduce a special class of relations with two arguments where the first argument is an entity of interest, such as `Bob Dylan` or `Michael Jordan` (the basketball player), and the second argument is a value of interest, such as their heights “171 cm” and “198 cm”, respectively.

The case for binary relations with values as second argument largely corresponds to the modeling of entity **attributes** in database terminology. Such relations are restricted to be *functions*: for each entity as first argument there is only one value for the second argument. We denote attributes in the same style as other relational properties, but we use numeric or string notation to distinguish the literals from entities:

⟨`Michael Jordan`, 198cm⟩ ∈ `height`	or
`height` (`Michael Jordan`, 198cm)	
⟨`Michael Jordan`, “Air Jordan”⟩ ∈ `nickname`	or
`nickname` (`Michael Jordan`, “Air Jordan”)	

Relations between Entities:
In addition to their attributes, entities are characterized by their **relationships** with other entities, for example, the birthplaces of people, prizes won, songs written or performed, and so on. Mathematical relations over classes, as introduced above, are the proper formalism for representing this kind of knowledge. The frequent case of **binary relations** captures the relationship between exactly two entities.

Some KBs focus exclusively on binary relations, and the Semantic Web data model RDF has specific terminology and formal notation for this case of so-called **subject-predicate-object triples**, or **SPO triples**, or merely **triples** for short.

The RDF model restricts the three roles in a **subject-predicate-object (SPO) triple** as follows:

- S must be a URI identifying an entity,
- P must be a URI identifying a relation, and
- O must be a URI identifying an entity for a relationship between entities, or a literal denoting the value of an attribute.

As binary relations can be easily cast into a labeled graph – with node labels for S and O and edge labels for P – knowledge bases that focus on SPO triples are widely referred to as **knowledge graphs**. SPO triples are often written in the form ⟨ `S, P, O` ⟩ or as `S P O` with the relation between subject and object. Examples of SPO triples are:

`Bob Dylan`	`married to`	`Sara Lownds`
`Bob Dylan`	`composed`	`Blowin’ in the Wind`
`Blowin’ in the Wind`	`composed by`	`Bob Dylan`
`Bob Dylan`	`has won`	`Nobel Prize in Literature`
`Bob Dylan`	`type`	`Nobel Laureate`

The examples also illustrate the notion of **inverse relations**: `composed by` is inverse to `composed`, and can be written also as $\texttt{composed}^{-1}$:

$$\langle S, O\rangle \in P \;\Leftrightarrow\; \langle O, S\rangle \in P^{-1}.$$

The last example in the above table shows that an entity belonging to a certain class can also be written as a binary relation, with **type** as the predicate, following the RDF standard. It also shows that knowledge can sometimes be expressed either by class membership or by a binary-relation property. In this case, the latter adds information (Nobel Prize in *Literature*) and the former is convenient for querying (about all Nobel Laureates). Moreover, having a class `Nobel Laureate` allows us to define further relations and attributes with this class as domain. To get the benefits of all this, we may want to have both of the example triples in the KB.

An advantage of binary relations is that they can express facts in a self-contained manner, even if some of the arguments for a higher-arity relation is missing or the instances of the relations are only partly known. For example, if we know only Dylan's birthplace but not his birthdate (or vice versa), capturing this in the ternary relation `birth` is a bit awkward as the unknown argument would have to be represented as a *null value* (i.e., a placeholder for an unknown or undefined value). In database systems, null values are standard practice, but they often make things complicated. In KBs, the common practice is to avoid null values and prefer binary relations where we can simply have a triple for the known argument (birthplace) and nothing else.

Higher-Arity Relations:
Some KBs emphasize binary relations only, leading to the notion of **knowledge graphs** (KGs). However, ternary and higher-arity relations can play a big role, and these cannot be directly captured by a graph.

At first glance, it may seem that we can always decompose a higher-arity relation into multiple binary relations. For example, instead of introducing the ternary relation $birth$: $person \times date \times city$, we can alternatively use two binary relations: $birthdate : person \times date$ and $birthplace : person \times city$. In this case, no information is lost by using simpler binary relations. Another case where such decomposition works well is the relation that contains all tuples of parents, sons and daughters: $children : person \times boys \times girls$. This could be equally represented by two separate relations $sons : person \times boys$ and $daughters : person \times girls$. In fact, database design theory tells us that this decomposition is a better representation, based on the notion of multi-valued dependencies [588].

However, not every higher-arity relation is decomposable without losing information. Consider a quarternary relation $won : person \times award \times year \times field$ capturing who won which prize in which year for which scientific field. Instances would include

⟨ `Marie Curie, Nobel Prize, 1903, physics` ⟩

and

$\langle$ `Marie Curie, Nobel Prize, 1911, chemistry` $\rangle$.

If we simply split these 4-tuples into a set of binary-relation tuples (i.e., SPO triples), we would end up with:

$\langle \texttt{MarieCurie}, \texttt{NobelPrize} \rangle$, $\langle \texttt{MarieCurie}, \texttt{1903} \rangle$, $\langle \texttt{MarieCurie}, \texttt{Physics} \rangle$,

$\langle \texttt{MarieCurie}, \texttt{NobelPrize} \rangle$, $\langle \texttt{MarieCurie}, \texttt{1911} \rangle$, $\langle \texttt{MarieCurie}, \texttt{Chemistry} \rangle$.

Leaving the technicality of two identical tuples aside, the crux here is that we can no longer reconstruct in which year Marie Curie won which of the two prizes. Joining the binary tuples using database operations would produce spurious tuples, namely, all four combinations of 1903 and 1911 with physics and chemistry.

The Semantic Web data model RDF and its associated W3C standards (including the SPARQL query language) support only binary relations. They therefore exploit clever ways of encoding higher-arity relations into a binary representation, based on techniques related to **reificiation** [603]. Essentially, each instance of the higher-arity relation is given an identifier of type `statement` and that identifier is combined with the original relation's arguments into a set of binary tuples.

For the n-ary relation instance $R(X_1, X_2 \ldots X_n)$ the **reified representation** consists of the set of binary instances

$type(id, statement)$, $arg_1(id, X_1), arg_2(id, X_2) \ldots arg_n(id, X_n)$

where id is an identifier.

With this technique, the triple $\langle$`id type statement`$\rangle$ asserts the existence of the higher-arity tuple, and the additional triples fill in the arguments. In some KBs, the technique is referred to as **compound objects**, as the $\langle$`id type statement`$\rangle$ is expanded into a set of *facets*, often called **qualifiers**, with the number of facets even being variable (see, e.g., [230]). Reification can be applied to binary relations as well (if desired): the representation of $\langle$`S P O`$\rangle$ then becomes $\langle$`id type statement`$\rangle$, $\langle$`id hasSubject S`$\rangle$, $\langle$`id hasPredicate P`$\rangle$, $\langle$`id hasObject O`$\rangle$. Our use case for reification is higher-arity relations, though, most importantly, to capture events and their different aspects. Attaching provenance or belief information to statements is another case for reification.

The identifier `id` could be a number or a URI (as required by RDF). The names of the facets of arguments arg_i ($i = 1..n$) can be arbitrarily chosen, but often capture certain properties that can be aptly reflected in their names. For example, a tuple for the higher-arity relation `wonAward` may result in the following triples:

id	type	statement
id	hasPredicate	wonAward
id	winner	Marie Curie
id	award	Nobel Prize
id	year	1903
id	field	physics

The example additionally includes a triple that encodes the name of the property `wonAward` for which the n-tuple holds. Strictly speaking, we could then drop the `id type statement` triple without losing information, and the remaining triples are a typical knowledge representation for n-ary predicates (e.g., for events): one triple for the predicate itself and one for each of the n-ary predicate arguments. If we want to emphasize two of the arguments as major subject and object, we could also use a hybrid form with triples like ⟨`id hasSubject Marie Curie`⟩, ⟨`id hasObject Nobel Prize`⟩, ⟨`id year 1903`⟩, ⟨`id field physics`⟩.

The advantage of the triples representation is that it stays in the world of binary relations, and the notion of a *knowledge graph* still applies. In a graph model, the SPO triple that encodes the existence of the original n-ary R instance is often called a *compound node* (e.g., in Freebase and in Wikidata), and serves as the "gateway" to the *qualifier triples*.

The downside of reification and related techniques for casting n-ary relations into RDF is that they make querying more difficult if not to say tedious. It requires more joins, and considering paths and not just single edges when dealing with compound nodes in the graph model. For this reason, some KBs have also pursued hybrid representations where for each higher-arity relation, the most salient pair of arguments are represented as a standard binary relation and reification is used only for the other arguments.

2.1.4 Canonicalization

An important objective for a clean knowledge base is the uniqueness of the subjects, predicates and objects in SPO triples and other relational statements. We want to capture every entity and every fact about it exactly once, just like an enterprise company should contain every customer and her orders and account balance once and only once. As soon as redundancy creeps in, this opens the door for variations of the same information and hence potential inconsistency. For example, if we include two entities in our KB, `Bob Dylan` and `Robert Zimmerman` (Dylan's real name) without knowing that they are the same, we could attach different facts to them that may eventually contradict each other. Furthermore, we would distort the results of counting queries (counting two people instead of one person). This motivates the following **canonicalization** principle:

> Each entity, class and property in a KB is **canonicalized** by having a unique identifier and being included exactly once.

For entities this implies the need for **named entity disambiguation**, also known as **entity linking** ([523]). For example, we need to infer that Bob Dylan and Robert Zimmerman are the same person and should have him as one entity with two different labels rather than two entities with different identifiers. The same principle should hold for classes, for example, avoiding that we have both `guitarists` and `guitar players`, and for properties as well.

We strive to avoid redundancy and the resulting ambiguities and potential inconsistencies. However, this goal is not always perfectly achievable in the entire life-cycle of KB creation, growth and curation. Some KBs settle for softer standards and allow diverse representations for the same facts to co-exist, effectively demoting entities and relations into literal values. Here is an example of such a softer (and hence less desirable, but still useful) representation:

Bob Dylan	has won	"Nobel Prize in Literature"
Bob Dylan	has won	"Literature Nobel Prize"
Bob Dylan	has won award	"Nobel"

2.1.5 Logical Invariants

In addition to the *grounded statements* about entities, classes and relational properties, KBs can also contain *intensional knowledge* in the form of logical constraints and rules. The purpose of **constraints** is to enforce the consistency of the KB: grounded statements that violate a constraint cannot be entered. For example, we do not allow a second birthdate for a person, as the `birthdate` property is a function, and we require creators of songs to be musicians (including composers and bands). The former is an example of a *functional dependency*, and the latter is an example of a *type constraint*. We discuss consistency constraints and their crucial role for KB curation in Chapter 8 (particularly, Section 8.3.1). The purpose of **rules** is to derive additional knowledge that logically follows from what the KB already contains. For example, if Bob is married to Sara, then Sara is married to Bob, by the symmetry of the `spouse` relation). We discuss rules in Section 8.3.2.

2.2 Design Space for Knowledge Base Construction

Machines cannot create any knowledge on their own; all knowledge about our world is created by humans (and their instruments) and documented in the form of encyclopedia, scientific publications, books, daily news, all the way to contents in online discussion forums and other social media. What machines actually do to construct a knowledge base is to tap into these sources, harvest their nuggets of knowledge, and refine and semantically organize them into a formal knowledge representation. This process of distilling knowledge from online contents is best characterized as **knowledge harvesting** [622, 621, 620].

This big picture opens up a design space for how we go about harvesting online contents towards large-scale knowledge bases. Depending on the kinds of sources we tap into and the standards that we set for the output we want to achieve, there is a variety of design choices. Figure 2.2 depicts this design space. Note that the connections between methods and their associated inputs and outputs are merely indicative for typical approaches; they are not meant to be exhaustive. For example, NLP tools and Deep Learning are useful for discovering entities and their types as well, but they face higher complexity and usually yield lower quality than the simpler methods based on rules and patterns.

Input Sources:
There is a wide spectrum of input sources to be considered. The top part of Figure 2.2 shows some notable design points, with difficulty increasing from left to right. The difficulties arise from the decreasing ratio of valuable knowledge to noise in the respective sources.

To build a high-quality KB, we advocate to start with the cleanest sources, called **premium sources** in the figure. These include well-organized and curated encyclopedic content like Wikipedia. For example, Wikipedia's set of article names is a great source for KB construction, as these names constitute the world's notable entities in reasonably standardized form: millions of entities with human-readable unique labels. First harvesting these entities (and cues about their classes, e.g., via Wikipedia categories) forms a strong

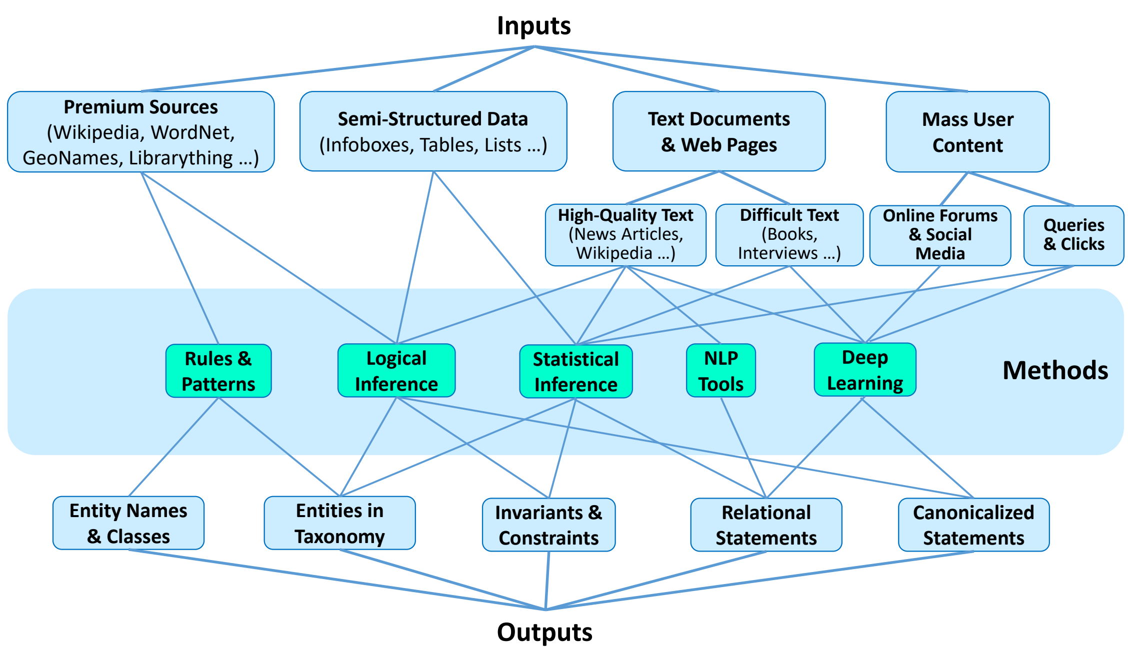


Figure 2.2: Design Space for Knowledge Harvesting

backbone for subsequent extensions and refinements. This design choice can be seen as an instantiation of the folk wisdom to "pick low-hanging fruit" first, widely applied in systems engineering. Beyond Wikipedia as a general-purpose source, knowledge harvesting should generally start with the most authoritative high-quality sources for the domains of interest. For example, for a KB about movies, it would be a mistake to disregard IMDB as it is the world's largest and cleanest – manually constructed – repository of movies, characters, actors, keywords and phrases about movie plots, etc. Likewise, we must not overlook sources like GeoNames and OpenStreetMap for geographic entities, GoodReads and Librarything for books, MusicBrainz (or even Spotify's catalog) for music, DrugBanks for medical drugs, and so on. Note that some sources may be proprietary and require licensing.

After considering premium sources, the next step is to tap into **semi-structured elements** in online data, like infoboxes (in Wikipedia and other wikis), tables, lists, headings, category systems, etc. This is almost always easier than comprehending textual contents. However, if we aim at rich coverage of facts about entities, we eventually have to extract knowledge from **natural-language text** as well – ranging from high-quality sources like Wikipedia articles all the way to user contents in online forums and other social media. Finally, mass-user data about online behavior – like queries and clicks – is yet another source, which comes with a large amount of noise and potential bias.

From these considerations it should be obvious that we generally face a **precision-recall trade-off**. In Figure 2.2, the precision of extracted knowledge tends to decrease from left to right, and the recall typically increases from left to right (with exceptions, though). That is, sources on the left end often yield highly accurate KBs but limited coverage, whereas sources on the right end usually yield less accurate KBs but with higher coverage. We define precision and recall as follows:

The **precision** of a KB of statements is the ratio

$$\frac{\#\ correct\ statements\ in\ KB}{\#\ statements\ in\ KB}.$$

The **recall** of a KB of statements is the ratio

$$\frac{\#\ correct\ statements\ in\ KB}{\#\ correct\ statements\ in\ real\ world}.$$

Precision can be evaluated by inspecting a KB alone, but recall can only be estimated as the complete real-world knowledge is not known to us. However, we can sample on a per-entity basis and compare what a KB knows about an entity (in the form of relational tuples) against what a human can learn from reading the Wikipedia article or other high-coverage sources about the entity.

Output Scope and Quality:
Depending on our goals on precision and recall of a KB and our choice on dealing with the trade-off, we can expect different kinds of outputs from a knowledge-harvesting machinery. Figure 2.2 lists major options in the bottom part.

The minimum that every KB-building endeavor should have is that all entities in the KB are semantically typed in a system of classes. Without any classes, the KB would merely be a flat collection of entities, and a lot of the mileage that search applications get from KBs is through the class system. For example, queries or questions about "singers who are also poets" can be answered by intersecting the entities of two classes (returning, e.g., Bob Dylan, Leonard Cohen, Serge Gainsbourg).

As for the entities themselves, some KBs do *not* normalize, lacking unique identifiers and including real-world duplicates. Such a KB of names, containing, for example, both Bob Dylan and Robert Zimmerman as if they were two entities (see Section 2.1.4), can still be useful for many applications. However, a KB with disambiguated names and canonicalized representation of entities clearly offers more value for high-quality use cases such as data analytics.

Larger coverage and application value come from capturing also properties of entities, in the form of relational statements. Often, but not necessarily, this goes hand in hand with logical invariants about properties, which could be acquired by rule mining or by hand-crafted modeling using expert or crowdsourced inputs. Analogously to surface-form versus canonicalized entities, relational statements also come in these two forms: with a diversity of names for the same logical relation, or with unique names and no redundancy (see Section 2.1.4 for examples).

Methodological Repertoire:
To go from input to output, knowledge harvesting has a variety of options for its algorithmic methods and tools. The following is a list of most notable options; there are further choices, and practical techniques often combine several of the listed options.

- **Rules and Patterns:** When inputs have rigorous structure and the desired output quality mandates conservative techniques, rule-based extraction can achieve best results. The System T project at IBM is a prime example of rule-based knowledge extraction for industrial-strength applications (see, e.g., [89]).
- **Logical Inference:** Using consistency constraints can often eliminate spurious candidates for KB statements, and deduction rules can generate additional statements of interest. Both cases require reasoning with computational logics. This is usually combined with other paradigms such as extraction rules or statistical inference.
- **Statistical Inference:** Distilling crisp knowledge from vague and ambiguous text content or semi-structured tables and lists often builds on the observation that there

is redundancy in content sources: the same KB statement can be spotted in (many) different places. Thus we can leverage statistics and corresponding inference methods. In the simplest case, it boils down to frequency arguments, but it can be much more elaborated by considering different statistical measures and joint reasoning. In particular, statistical inference can be combined with logical invariants, for example, by probabilistic graphical models ([126]).

- **NLP Tools:** Modern tools for natural language processing (see, e.g., [146, 273]) encompass a variety of methods, from rule-based to deep learning. They reveal structure and text parts of interest, such as dependency-parse trees for syntactic analysis, pronoun resolution, identification of entity names, sentiment-bearing phrases, and much more. However, as language becomes more informal with incomplete sentences and colloquial expressions (incl. social-media talks such as "LOL"), mainstream NLP does not always work well.

- **Deep Learning:** The most recent addition to the methodogical repertoire is deep neural networks, trained in a supervised or distantly supervised manner (see, e.g., [68, 186]). The sweet spot here is when there is a large amount of "gold-standard" labeled training data, and often in combination with learning so-called *embeddings* from large text corpora. Thus, deep learning is most naturally used for increasing the coverage of a KB after initial population, such that the initial KB can serve as a source of distant supervision.

In Figure 2.2, the edges between input and methods and between outputs and methods indicate choices for methods being applied to different kinds of inputs and outputs. Note that this is not meant to exclude further choices and additional combinations.

The outlined design space and the highlighted options are by no means complete, but merely reflect some of the prevalent choices as of today. We will largely use this big picture as a "roadmap" for organizing material in the following chapters. However, there are further options and plenty of underexplored (if not unexplored) opportunities for advancing the state of the art in knowledge harvesting.

3 Knowledge Integration from Premium Sources

This chapter presents a powerful method for populating a knowledge base with entities and classes, and for organizing these into a systematic taxonomy. This is the backbone that any high-quality KB – broadly encyclopedic or focused on a vertical domain – must have. Following the rationale of our design-space discussion, we focus here on knowledge harvesting from premium sources such as Wikipedia or domain-specific repositories such as GeoNames for spatial entities or GoodReads and Librarything for the domain of books. This emphasizes the philosophy of "picking low-hanging fruit first" for the best benefit/cost ratio.

3.1 The Case for Premium Sources

We recommend to start every KB construction project by tapping one or a few premium sources first. Such sources should have the following characteristics:

- authoritative high-quality content about entities of interest,
- high coverage of many entities, and
- clean and uniform representation of content, like having clean HTML markup or even wiki markup, unified headings and structure, well-organized lists, informative categories, and more.

Distilling some of the contents from such sources into machine-readable knowledge can create a strong *core KB* with a good ratio of "mileage per effort". In particular, relatively simple extraction and cleaning methods go a long way already. The core KB can then be further expanded from other sources – with more advanced methods as presented in subsequent chapters.

Wikipedia:
For a general-purpose encyclopedic knowledge base, **Wikipedia** is presumably the most suitable starting point, with its huge number of entities, highly informative descriptions and annotations, and quality assurance via curation by a large community and a sophisticated system of moderators. The English edition of Wikipedia (https://en.wikipedia.org) contains more than 6 million articles with 500 words of text on average (as of July 1, 2020), all with unique names most of which feature individual entities. These include more than 1.5 million notable people, more than 750,000 locations of interest, more than 250,000 organizations, and instances of further major classes including events (i.e., sports tournaments, natural disasters, battles and wars, etc.) and creative works (i.e., books, movies, musical pieces, etc.).

Another great starting point, with even more entities (ca. 100 million), would be the **Wikidata** knowledge base (https://wikidata.org), populated with entity-centric facts (SPO triples) by a knowledge-sharing community [600]. Wikidata is already a full-fledged KB, in a formal representation following the RDF data model. So there is no point in illustrating knowledge extraction from Wikidata. Moreover, Wikidata largely focuses on capturing basic biographic facts about entities, like birthdate, birthplace, spouses, children for people, or city, country and geo-coordinates for buildings and landmarks, and so on. Its type system is large, but most entities belong only to one or two types, whereas Wikipedia often offers several tens of highly informative categories that characterize an entity. Last but not least, Wikidata does not have full-text articles about entities with rich descriptions, lists, tables and more. We will come back to Wikidata in Chapter 9, including a case for integration with another knowledge source (see Section 9.1).

Wikipedia serves as an archetype of knowledge-sharing communities, which can be seen as "proto-KBs": the right contents for a KB, but not yet in the proper representation. Another case in point would be the Chinese encyclopedia Baidu Baike with almost 20 million articles (https://baike.baidu.com). In this chapter, we focus on the English Wikipedia as an exemplary case. We will see that Wikipedia alone does not lend itself to building a clean KB as easily as one would hope. Therefore, we combine input from Wikipedia with another premium source: the **WordNet** lexicon [159] as a key asset for the KB taxonomy.

Geographic Knowledge:
For building a KB about geographic and geo-political entities, like countries, cities, rivers, mountains, natural and cultural landmarks, Wikipedia itself is a good starting point, but there are very good alternatives as well. Wikivoyage (https://www.wikivoyage.org) is a travel-guide wiki with specialized articles about travel destinations. GeoNames (https://www.geonames.org) is a huge repository of geographic entities, from mountains and volcanos to churches and city parks, more than 10 million in total. If city streets, highways, shops, buildings and hiking trails are of interest, too, then OpenStreetMaps (https://www.openstreetmap.org/) is another premium source to consider (or alternatively commercial maps if you can afford to license them). Even commercial review forums such as TripAdvisor (https://www.tripadvisor.com/) could be of interest, to include hotels, restaurants and tourist services.

These sources complement each other, but they also overlap in entities. Therefore, simply taking their union as an entity repository is not a viable solution. Instead, we need to carefully integrate the sources, using techniques for *entity matching* to avoid duplicates and to combine their different pieces of knowledge for each entity (see Chapter 5, especially Section 5.2).

To obtain an expressive and clean taxonomy of classes, we could tap each of the sources separately, for example, by interpreting categories as semantic types. But again, simply taking a union of several category systems does not make sense. Instead, we need to find ways of aligning equivalent (and possibly also subsumption) pairs of categories, as a basis for constructing a *unified type hierarchy*. For example, can and should we map `craters` from one source to `volcanoes` in a second source, and how are both related to `volcanic national parks`? This alignment and integration is not an easy task, but it is still much simpler than extracting all volcanoes and craters from textual contents in a wide variety of diverse web pages.

Knowledge about Movies:
For the movie domain, we are primarily interested in entities like movies, directors, actors, producers, soundtrack music, contributors to special effects etc. IMDB (Internet Movie Database, https://www.imdb.com/) is by far the best source of information for this scope. This premium source is commercial and disallows crawling, but it offers periodic dumps of its core data for downloading (subject to licensing conditions).

However, advanced users may ask for more: a movie KB should also provide convenient access to additional knowledge about the life of the movie contributors, for example, how often they are divorced and how they started their careers. To this end, we could combine IMDB entities with selected articles from *Wikipedia* or entries from *Wikidata*, but such combinations involve non-trivial knowledge integration tasks. Moreover, although IMDB is huge, it does not have perfect coverage of the world's film footage, for example, missing many Bollywood productions, African movies or lesser-known documentaries. Therefore, merging its repository with entities from other sites would be desirable, with appropriate *entity matching* and *type alignment*, similar to the Wikipedia case discussed earlier.

An even better KB for movie afficionados would cover also the contents of movies and their characters. A rich source about popular movies and TV series is *fan-community wikis*, many of which are hosted at https://www.fandom.com/ (formerly registered as https://www.wikia.com/). On these wikis, a large number of fictitious characters are systematically organized in semantic types, and are annotated with crisp statements about their traits and relationships in the respective stories. The type labels allow finding favorite villains, heroes, wizards, witches, and more. However, if we want a truly integrated and clean KB, we need to align these types with the categories that IMDB provides for real people and some movie characters, for example, to consider the gender and race of actors and their roles.

Health Knowledge:
Another vertical domain of great importance for society is health: building a KB with entity instances of diseases, symptoms, drugs, therapies etc. There is no direct counterpart to Wikipedia for this case, but there are large and widely used tagging catalogs and terminology lexicons like MeSH (https://www.nlm.nih.gov/mesh/meshhome.html) and UMLS (https://www.nlm.nih.gov/research/umls/ including the SNOMED clinical terminology for healthcare), and these can be treated as analogs to Wikipedia: rich categorization but not always semantically clean. The next step would then be to clean these raw assets, using methods like the presented ones, and populate the resulting classes with entities.

For the latter, additional premium sources could be considered: either Wikipedia articles about biomedical entities, or curated structured sources such as DrugBank (https://www.drugbank.ca/) or Disease Ontology (http://disease-ontology.org/), and also human-oriented Web portals like the one by the Mayo Clinic (https://www.mayoclinic.org/patient-care-and-health-information).

Research projects along the lines of this knowledge integration and taxonomy construction for health include KnowLife/DeepLife [150, 148], Life-INet [487] and Hetionet [234]; see also [270] for general discussion of health knowledge.

3.2 Category Cleaning

Many premium sources come with a rich **category system**: assigning pages to relevant categories that can be viewed as proto-classes but are too noisy to be considered as a semantic type system. Wikipedia, as our canonical example, organizes its articles in a hierarchy of more than 1.9 million categories (as of July 1, 2020). For example, `Bob Dylan` (the entity corresponding to article en.wikipedia.org/wiki/Bob_Dylan) is placed in categories such as

`American male guitarists`,
`Pulitzer Prize winners`,
`Songwriters from Minnesota` etc.,

and `Blowin' in the Wind` (corresponding to en.wikipedia.org/wiki/Blowin'_in_the_Wind) is in categories such as

`Songs written by Bob Dylan`,
`Elvis Presley songs`,
`Songs about freedom` and
`Grammy Hall of Fame recipients`, among others.

Using these categories as classes with their respective entities, it seems we could effortlessly construct an initial KB. So are we done already?

Unfortunately, the Wikipedia category system is almost a class taxonomy, but only almost. We face the following difficulties:

- **High Specificity of Categories:** The direct categories of entities (i.e., leaves in the category hierarchy) tend to be highly specific and often combine multiple classes into one multi-word phrase. Examples are `American male singer-songwriters`, `20th-Century American guitarists` or `Nobel laureates absent at the ceremony`. For humans, it is obvious that this implies membership in classes `singers`, `men`, `guitar players`, `Nobel laureates` etc., but for a computer, the categories are initially just noun phrases.
- **Drifting Super-Categories:** By considering also super-categories (i.e., non-leaf nodes in the hierarchy) and the paths in the category system, we could possibly generalize the leaf categories and derive broader classes of interest, such as `men`, `American people`,

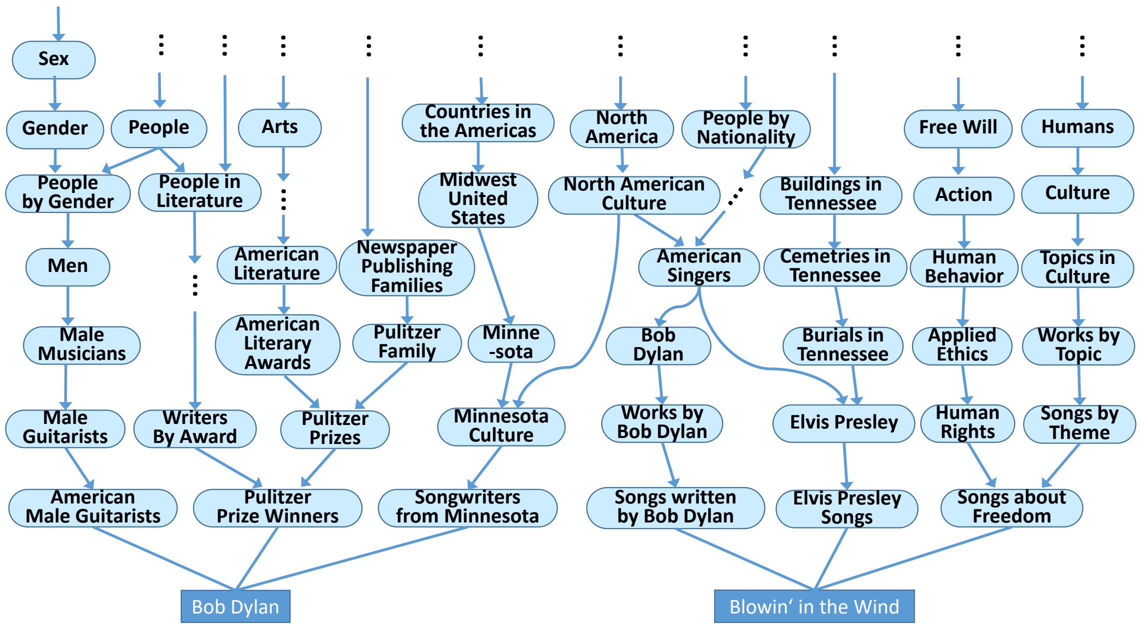


Figure 3.1: Example categories and super-categories from Wikipedia

`musicians`, etc. However, the Wikipedia category system exhibits conceptual drifts where super-categories imply classes that are incompatible with those of the corresponding leaves and the entity itself. Figure 3.1 shows excerpts of the category hierarchy for the entities `Bob Dylan` and `Blowin' in the Wind`. By transitivity, the super-categories would imply that Bob Dylan is a location, a piece of art, a family and a kind of reproduction process (generalizing the "sex" category). For the example song, the category system alone would likewise lead to flawed or meaningless classes: locations, buildings, singers, actions, etc.

- **Entity Types vs. Associative Categories:** Some of the super-categories are general concepts, for example, `Applied Ethics` and `Free Will` in Figure 3.1. Some of the edges between a category and its immediate super-category are conceptual leaps, for example, moving from songs and works to the respective singers in Figure 3.1.

All this makes sense when the category hierarchy is viewed as a means to support user browsing in an associative way, but it is not acceptable for the taxonomic backbone of a clean KB. For example, queries about buildings on North America may return `Blowin' in the Wind` as an answer, and statistical analytics on prizes, say by geo-region or gender, would confuse the awards and the awardees.

In the following we present a simple but powerful methodology to leverage the Wikipedia categories as raw input with thorough cleansing of their noun-phrase names and integration with upper-level taxonomies for clean KB construction based on the works of [456, 457, 562, 240] (applied in the WikiTaxonomy and YAGO projects).

Head Words of Category Names:
As virtually all category names are noun phrases, a basic building block for uncovering their semantics is to parse these multi-word phrases into their syntactic constituents. This task is known as *noun-phrase parsing* in NLP (see, e.g., [273] and [146]). In general, noun phrases consist of nouns, adjectives, determiners (like "the" or "a"), pronouns, coordinating conjunctions ("and" etc.), prepositions ("by", "from" etc.), and possibly even further word variants.

A typical first step is to perform **part-of-speech tagging**, or **POS tagging** for short. This tags each word with is syntactic sort: noun, adjective etc. Nouns are further classified into common nouns, which can have an article, e.g., "guitarist", and proper nouns which denote names (e.g., "Minnesota"). POS tagging usually works by dynamic programming over a pre-trained statistical model of word-variant sequences in large corpora. The subsequent **noun-phrase parsing** computes a syntactic tree structure for the POS-tagged word sequence, inferring which word modifies or refines which other word. This is usually based on stochastic context-free grammars, again using some form of dynamic programming. Later chapters in this article will make intensive use of such NLP methods, too.

The root of the resulting tree is called the **head word**, and this is what we are mostly after. For example, for "American male guitarists" the head word is "guitarists" and for "songwriters from Minneota" it is "songwriters". The head word is preceded by so-called *pre-modifiers*, and followed by *post-modifiers*. Sometimes, words from these modifiers can be combined with the head word to form a semantically meaningful class as well (e.g., "female guitarists").

Equipped with this building block, we can now tackle the desired category cleaning. Our goal is to distinguish taxonomic categories (such as "guitarists") from associative categories (such as "music"). The key idea is the heuristics that plural-form common nouns are likely to denote classes, whereas single-form nouns tend to correspond to general concepts (e.g., "free will"). The reason for this is that classes regroup several instances, and a plural form is thus a strong indicator for a class. We can possibly relax this heuristics to consider also singular-form nouns where the corresponding plural form is frequently occurring in corpora such as news. For example, if a Wikipedia category were named "jazz band" rather than "jazz bands" we should accept it as a class, while still disregarding categories such as "free will" or "ethics" (where "wills" is very rare, and "ethics" is a singular word despite ending with "s"). These ideas can be cast into the following algorithm.

Algorithm for Category Cleaning
Input: Wikipedia category name c: leaf node or non-leaf node
Output: semantic class label or `null`

1. Run noun-phrase parsing to identify headword h and modifier structure: $c = pre_1 \,..\, pre_k\ h\ post_1 \,..\, post_l$.
2. Test if h is in plural form or has a frequently occurring plural form. If not, return `null`.
   Optionally, consider also $pre_i \,..\, pre_k\ h$ as class candidates, with increasing i from 0 to $k-1$.
3. For leaf category c, return h (and optionally additonal class labels $pre_i \,..\, pre_k\ h$).
4. For non-leaf category c, test if the class candidate h is a synonym or hypernym (i.e., generalization) of an already accepted class (including h).
   If so, keep it; otherwise, discard it.

The rationale for the additional test in Step 4 is that non-leaf categories in Wikipedia are often merely associative, as opposed to denoting semantically proper super-classes (see discussion above). So we impose this additional scrutinizing, while still being able to harvest the cases when head words of super-categories are meaningful outputs (e.g., `Musicians` and `Men` in Figure 3.1). The test itself can be implemented by looking up head words in existing dictionaries like **WordNet** [159] or **Wiktionary** (https://www.wiktionary.org/), which list synonyms and hypernyms for many words. This is a prime case of harnessing a second premium source.

Class Candidates from Wikipedia Articles:
We have so far focused on Wikipedia *categories* as a source of semantic class candidates. However, normal Wikipedia *articles* may be of interest as well. Most articles represent individual entities, but some feature concepts, among which some may qualify as classes. For example, the articles https://en.wikipedia.org/wiki/Cover_version and https://en.wikipedia.org/wiki/Aboriginal_Australians correspond to classes as they have instances, whereas https://en.wikipedia.org/wiki/Empathy is a singleton concept as it has no individual entities as instances of interest.

Simple but effective heuristics to capture these cases have been studied by [198, 440]. The key idea is that an article qualifies as a class if its textual body mentions the article's title in both singular and plural forms. For example, "cover version" and "cover versions" are both present in the article about cover versions (of songs), but the article on empathy does not refer to "empathys". Obviously, this technique is just another building block that should be combined with other heuristics and statistical inference.

3.3 Alignment between Knowledge Sources

By applying the category cleaning algorithm to all Wikipedia categories, we can obtain a rich set of class labels for each entity. However, as the Wikipedia community does not enforce strict naming standards, we could arrive at duplicates for the same class, for example, accepting both `guitarist` and `guitar player`. Moreover, as we are mostly harvesting the leaf-node categories and expect to prune many of the more associative super-categories, our KB taxonomy may end up with many disconnected classes.

To fix these issues, we resort to pre-existing high-quality taxonomies like **WordNet** [159]. This lexicon already covers more than hundred thousand concepts and classes – called *word senses* or *synsets* for sets of synonyms – along with clean structure for hypernymy. Alternatively, we could consider **Wiktionary** (https://www.wiktionary.org/). Both of these lexical resources also have multilingual extensions, covering a good fraction of mankind's languages. See also [203] for a general overview of this kind of lexical resources.

A major caveat, however, is that WordNet has hardly any entity-level instances for its classes; you can think of it as an un-populated *upper-level taxonomy.* The same holds for Wiktionary. The goal now is to align the class candidates harvested from Wikipedia with the classes in WordNet.

Similarity between Categories and Classes:
The key idea is to perform a similarity test between a class candidate from Wikipedia and potentially corresponding classes in WordNet. In the simplest case, this is just a surface-form **string similarity**. For example, the Wikipedia-derived candidate "guitar player" has high similarity with the WordNet entry "guitarist". There are two problems to address, though. First, we could still observe low string similarity for two matching classes, for example, "award" from Wikipedia against "prize" in WordNet. Second, we can find multiple matches with high similarity, for example "building" from Wikipedia matching two different senses in WordNet, namely, building in the sense of a man-made structure (e.g., houses, towers etc.) and building in the sense of a construction process (e.g., building a KB). We have to make the right choice among such alternatives for ambiguous words.

The solution for the first problem – similarity-based matching – is to consider contexts as well. WordNet, and Wiktionary alike, provide synonym sets as well as short descriptions (so-called glosses), for their entries, and the Wikipedia categories can be contextualized by related words occurring in super-categories or (articles for) their instances. This way, we are able to map "award" to "prize" because WordNet has the entry "prize, award (something given for victory or superiority in a contest or competition or for winning a lottery)", stating that "prize" and "award" are synonyms (for this specific word sense). More generally, we could consider also entire neighborhoods of WordNet entries defined by hypernyms, hyponyms, derivationally related terms, and more. Such contextualized **lexical similarity** measures have been investigated in research on **word sense disambiguation (WSD)**, see [417, 203] for overviews.

Another approach to strengthen the similarity comparisons is to incorporate **word embeddings** such as Word2Vec [384] or Glove [447] (or even deep neural networks along these lines, such as BERT [118]). We will not go into this topic now, but will come back to it in Section 4.5.

For the second problem – ambiguity – we could apply state-of-the-art WSD methods, but it turns out that there is a very simple heuristic that works so well that it is hardly outperformed by any advanced WSD method. It is known as the **most frequent sense (MFS)** heuristic: whenever there is a choice among different word senses, pick the one that is more frequently used in large corpora such as news or literature. Conveniently, the WordNet team has already manually annotated large corpora with WordNet senses, and has recorded the frequency of each word sense. It is thus easy to identify, for each given word, its most frequent meaning. For example, the MFS for "building" is indeed the man-made structure. There are exceptions to the MFS heuristics, but they can be handled in other ways.

Putting Everything Together:
Putting these considerations together, we arrive at the following heuristic algorithm for aligning Wikipedia categories and WordNet senses.

Algorithm for Alignment with WordNet
Input: Class name c derived from Wikipedia category
Output: synonym or hypernym in WordNet, or `null`

1. Compute string or lexical similarity of c to WordNet entries s (for candidates s with certain overlap of character-level substrings). Then pick the s with highest similarity if this is above a given threshold; otherwise return `null`.
2. If the highest-similarity entry s is unambiguous (i.e., the same word has only this sense), then return the WordNet sense for s. If s is an ambiguous word, then return the MFS for s (or use another WSD method for c and the s candidates).

Once we have mapped the accepted Wikipedia-derived classes onto WordNet, we have a complete taxonomy, with the upper-level part coming from the clean WordNet hierarchy of hypernyms. The last thing left to decide for the alignment task is whether the category-based class c is synonymous to the identified WordNet sense s or whether s is a hypernym of c. The latter occurs when c does not have a direct counterpart in WordNet at all. For example, we could keep the category $c =$ "singer-songwriter", but WordNet does not have this class at all. Instead we should align c to `singer` or `songwriter` or both. If WordNet did not have an entry for songwriters, we should map c to the next hypernym, which is `composer`.

This final issue can be settled heuristically, for example, by assuming a synonymy match if the similarity score is very high and assuming a hypernym otherwise, or we could resort to leveraging information-theoretic measures over additional text corpora (e.g., the full text of all Wikipedia articles). Specifically, for a pair c and s (e.g., `songwriter` and `composer` – a case of hypernymy), a symmetry-breaking measure is the conditional probability $P[s|c]$ estimated from co-occurrence frequencies of words. If $P[s|c] \gg P[c|s]$, that is, c sort of implies s but not vice versa, then s is likely a hypernym of c, not a synonym. Various measures along these lines are investigated in [618, 183].

The presented methodology can also be adapted to other cases of taxonomy alignment, for example, to GeoNames, Wikivoyage and OpenStreetMap (and Wikipedia or Wikidata) about geographic categories and classes (see Section 3.1).

3.4 Graph-based Alignment

A different paradigm for aligning Wikipedia categories with WordNet classes, as prime examples of premium sources, has been developed by [418] for constructing the *BabelNet* knowledge base. It is based on a candidate graph derived from the Wikipedia category at hand and potential matches in WordNet, and then uses graph metrics such as shortest paths or graph algorithms like random walks to infer the best alignment. Figure 3.2 illustrates this approach. The graph construction extracts the head word of interest and salient context words from Wikipedia – "play" as well as "film" and "fiction" in the example. Then all approximate matches are identified in WordNet, and their respective local neighborhoods are added to the graph casting WordNet's lexical relations like hypernymy/hyponymy, holoynymy/meronymy (whole-part) etc. into edges. Edges could even be weighted based on similarity or salience metrics.

In the example, we have two main candidates to which "play" could refer: "play, drama" or "play (sports)" (WordNet contains even more). To rank these candidates, the simplest method is to look at how close they are to the Wikipedia-based start nodes "play", "film" and "fiction", for example, by aggregating the shortest paths from start nodes to candidate-class nodes. Alternatively, and typically better performing, we can use methods based on **random walks** over the graph, analogously to how Google's original PageRank and PersonalizedPageRank measures were computed ([59, 263]).

The walk starts at the Wikipedia-derived head word "play" and randomly traverses edges to visit other nodes – where the probabilities for picking edges should be proportional to edge weights (i.e., uniform over all outgoing edges if the edges are unweighted). Occasionally, the walk could jump back to the start node "play", as decided by a probabilistic coin toss. By repeating this random procedure sufficiently often, we obtain statistics about how often each node is visited. This statistics converges to well-defined *stationary visiting probabilities* as the walk length (or repetitions after jumping back to the start node) approaches infinity.

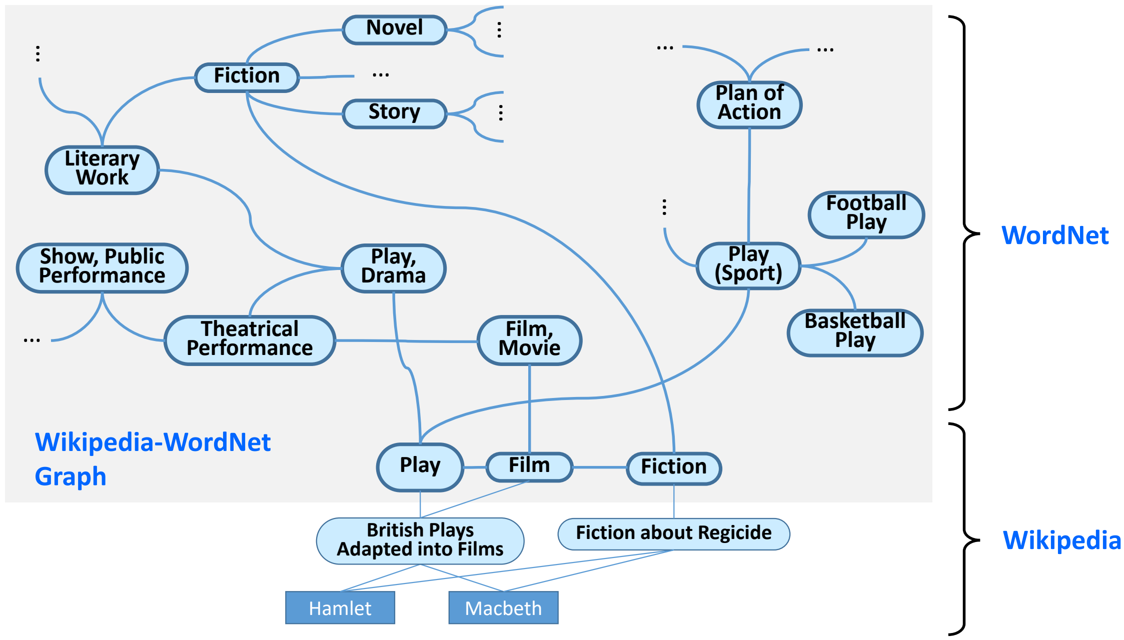


Figure 3.2: Example for graph-based alignment

The candidate class with the highest visiting probability is the winner: "play, drama" in the example. Such random-walk methods are amazingly powerful, easy to implement and widely applicable. We will see other use cases in later chapters.

3.5 Discussion

There are various extensions of the presented method. Wikipedia category names do not just indicate class memberships, but often reflect other relations as well. For example, for Bob Dylan being in the category `Nobel Laureates in Literature` we can infer a relational triple ⟨ `Bob Dylan`, `has won`, `Literature Nobel Prize` ⟩. Such extensions have been developed by [415]; we will revisit these techniques in Chapter 6. Another line of generalization is to leverage the taxonomies from the presented methods as training data to learn generalized patterns in category names and other Wikipedia structures (infobox templates, list pages etc.). This way, the taxonomy can be further grown and refined. Such methods have been developed in the Kylin/KOG project by [626, 625].

Relationship with Ontology Alignment:
The alignment between Wikipedia categories and WordNet classes can be seen as a special case of **ontology alignment** (see, e.g., [551] for an overview, [20, 331] for representative

state-of-the-art methods, and http://oaei.ontologymatching.org/ for a prominent benchmark series). Here, the task is to match classes and properties of one ontology with those of a second ontology, where the two ontologies are given in crisp logical forms like RDFS schemas or, even better, in the OWL description-logic language [601]. Ontology matching is in turn highly related to the classical task of **database schema matching** [123].

The case of matching Wikipedia categories and WordNet classes is a special case, though, for two reasons. First, WordNet has hardly any instances of its classes, and we chose to ignore the few existing ones. Second, the upper part of the Wikipedia category hierarchy is more associative than taxonomic so that it had to be cleaned first (as discussed in Section 3.2). For these reasons, the case of Wikipedia and WordNet benefits from tailored alignment methods, and similar situations are likely to arise also for domain-specific premium sources.

Beyond Wikipedia:
We used Wikipedia and WordNet as exemplary cases of premium sources, and pointed out a few vertical domains for wider applicability of the presented methods. Aligning and enriching pre-existing knowledge sources is also a key pillar for industrial-strength KBs about retail products (see, e.g., [117, 133]). More discussion on this use case is offered in Section 9.5.

Apart from this mainstream, similar cases for knowledge integration can be made for less obvious domains, too, examples being food [15, 218] or fashion [277, 189]. Food KBs have integrated sources like the FoodOn ontology [136], the nutrients catalog https://catalog.data.gov/dataset/food-and-nutrient-database-for-dietary-studies-fndds and a large recipe collection [369], and fashion KBs could make use of contents from catalogs such as https://ssense.com. More exotic verticals to which the Wikipedia-inspired methodology has been carried over are *fictional universes* such as Game of Thrones, the Simpsons, etc., with input from rich category systems by fan-community wikis (https://www.fandom.com/). Recent research on this topic includes works by [231] and [97]. Finally, another non-standard theme for KB construction is *how-to knowledge*: organizing human tasks and procedures for solving them in a principled taxonomy. Research on this direction includes [641, 98].

3.6 Take-Home Lessons

We summarize this chapter by the following take-home lessons.

- For building a core KB, with individual entities organized into a clean taxonomy of semantic types, it is often wise to start with one or a few *premium sources*. Examples are Wikipedia for general-purpose encyclopedic knowledge, GeoNames and WikiVoyage for geo-locations, or IMDB for movies.
- A key asset are categories by which entities are annotated in these sources. As categories are often merely associative, designed for manual browsing, this typically involves a

category cleaning step to identify taxonomically clean classes.

- To construct an expressive and clean taxonomy, while harvesting two or more premium sources, it is often necessary to integrate different type systems. This can be achieved by *alignment heuristics* based on NLP techniques (such as noun phrase parsing), or by *random walks* over candidate graphs.

4 KB Construction: Entity Discovery and Typing

This chapter presents advanced methods for populating a knowledge base with entities and classes (aka. types), by tapping into textual and semi-structured sources. Building on the previous chapter's insights on harvesting premium sources first, this chapter extends the regime for inputs to discovering entities and type information in Web pages and text documents. This will often yield noisier output, in the form of entity duplicates. The following chapter, Chapter 5, will address this issue by presenting methods for canonicalizing entities into unique subjects in the KB.

4.1 Problem and Design Space

Harvesting entities and their classes from premium sources goes a long way, but it is bound to be incomplete when the goal is to fully cover a certain domain such as music or health, and to associate all entities with their relevant classes. Premium sources alone are typically insufficient to capture long-tail entities, such as less prominent musicians, songs and concerts, as well as long-tail classes such as left-handed cello players or cover songs in a different language than the original. In this section, we present a suite of methods for automatically extracting such additional entities and classes from sources like web pages and other text documents.

In addressing this task, we typically leverage that premium sources already give us a taxonomic backbone populated with prominent entities. This leads to various discovery tasks:

1. Given a class T containing a set of entities $E = \{e_1 \dots e_n\}$, find more entities for T that are not yet in E.
2. Given an entity e and its associated classes $T = \{t_1 \dots t_k\}$, find more classes for e not yet captured in T.
3. Given an entity e with known names, find additional (alternative) names for e, such as acronyms or nicknames. This is often referred to as *alias name discovery*.
4. Given a class t with known names, find additional names for t. This is sometimes referred to as *paraphrase discovery*.

In the following, we organize different approaches by methodology rather than by these four tasks, as most methods apply to several tasks.

In principle, there is also a case where the initial repository of entities and classes is empty – that is, when there is no premium source to be harvested first. This is the case for *ab-initio taxonomy induction* from noisy observations, which will be discussed in Section 4.6.

4.2 Dictionary-based Entity Spotting

An important building block for all of the outlined discovery tasks is to detect **mentions** of already known entities in web pages and text documents. These mentions do not necessarily take the form of an entity's full name as derived from premium sources. For example, we want to be able to spot occurrences of `Steve Jobs` and `Apple Inc.` in a sentence such as "Apple co-founder Jobs gave an impressive demo of the new iPhone." Essentially, this is a string matching task where we compare known names of entities in the existing KB against textual inputs. The key asset here is to have a rich **dictionary of alias names** from the KB. Early works on information extraction from text made extensive use of name dictionaries, called *gazetteers*, in combination with NLP techniques (POS tagging etc.) and string patterns. Two seminal projects of this sort are the *GATE* toolkit [106, 107] and the *UIMA* framework [164].

A good dictionary should include

- abbreviations (e.g., "Apple" instead of "Apple Inc."),
- acronyms (e.g., "MS" instead of "Microsoft"),
- nicknames and stage names (e.g., "Bob Dylan" vs. his real name "Robert Zimmerman", or "The King" for "Elvis Presley"),
- titles and roles (e.g., "CEO Jobs" or "President Obama"), and possibly even
- rules for deriving short-hand names (e.g., "Mrs. Y" for female people with last name "Y").

Where do we get such dictionaries from? This is by itself a research issue, tackled, for example, by [76]. The easiest approach is to exploit redirects in premium sources and hyperlink anchors. Whenever a page with name X is redirected to an official page with name Y and whenever a hyperlink with anchor text X points to a page Y, we can consider X an alias name for entity Y. In Wikipedia, for example, a page with title "Elvis" (https://en.wikipedia.org/w/index.php?title=Elvis) redirects to the proper article (https://en.wikipedia.org/wiki/Elvis_Presley), and the page about the Safari browser contains a hyperlink with anchor "Apple" that points to the article https://en.wikipedia.org/wiki/Apple_Inc.. This approach extends to links from Wikipedia disambiguation pages: for example, the page https://en.wikipedia.org/wiki/Robert_Zimmerman lists 11 people with this name, including a link to https://en.wikipedia.org/wiki/Bob_Dylan. Of course, this does not resolve the ambiguity of the name, but it gives us one additional alias name for `Bob Dylan`. Hyperlink anchor texts in Wikipedia were first exploited for entity alias names by [67], and this simple technique was extended to hyperlinks in arbitrary web pages by [550]. A special case of interest is to harvest multilingual names from interwiki links in Wikipedia (connecting different language editions) or non-English web pages linking to English Wikipedia articles. For example, French pages with anchor text "Londre" link to the article https://en.wikipedia.org/wiki/London. All this is low-hanging fruit from an engineering perspective, and gives great mileage towards rich dictionaries for entity names.

An analogous issue arises for class names as well. Here, redirect and anchor texts are useful, too, but are fairly sparse. The WordNet thesaurus and the Wiktionary lexicon contain synonyms for many word senses (e.g., “vocalists” for `singers`, and vice versa) and can serve to populate a dictionary of class paraphrases.

The ideas that underlie the above heuristics can be cast into a more general principle of **strong co-occurrence**:

Strong Co-Occurrence Principle:
If an entity or class name X co-occurs with name Y in a context with cue Z, then Y is (likely) an alias name for X.

This principle can be instantiated in various ways, depending on what we consider as context cue Z:

- The cue Z is a hyperlink where X is the link target and Y is the anchor text.
- The cue Z is a specific wording in a sentence, like “also known as”, “aka.”, “born as”, “abbreviated as”, “for short” etc.
- The context cue Z is a query-click pair (observed by a search engine), where X is the query and Y is the title of a clicked result (with many clicks by different users).
- The context cue Z is the frequent occurrence of X in documents about Y (e.g., Wikipedia articles, biographies, product reviews etc.).

For example, when many users who query for “Apple” (or “MS”) subsequently click on the Wikipedia article or homepage of `Apple Inc.` (or “Microsoft”), we learn that “Apple” is a short-hand name for the company. This co-clicking technique has been studied by [577]. Extending this to textual co-occurrence (i.e., the last of the above itemized cases) comes with a higher risk of false positives, but could still be worthwhile for cases like short names or acronyms for products. The technique can be tuned towards either precision or recall by thresholding on the observation frequencies and by making the context cue more or less restrictive. More advanced techniques for learning co-occurrence cues about alias names – so-called synonym discovery, have been investigated by [469], among others.

4.3 Pattern-based Methods

4.3.1 Patterns for Entity-Type Pairs

Hearst Patterns:
To discover more entities of a given class or more classes of a given entity, a powerful

approach is to consider specific patterns that co-occur with input (class or entity) and desired output (entity and class). For example, a text snippet like "singers such as Bob Dylan, Elvis Presley and Frank Sinatra" suggests that Dylan, Presley and Sinatra belong to the class of `singers`. Such patterns have been identified in the seminal work of Marti Hearst [224], and are thus known as **Hearst patterns**. They are a special case of the strong co-occurrence principle where the Hearst patterns serve as context cues.

In addition to the "such as" pattern, the most important Hearst patterns are: "X like Y" (with class X in plural form and entity Y), "X and other Y" (with entity X and class Y in plural form), and "X including Y" (with class X in plural form and entity Y).

Some of the Hearst patterns also apply to discovering subclass relations between classes. In the pattern "X including Y", X and Y could both be classes, for example, "singers including rappers". In fact, the patterns alone cannot distinguish between observations of entity-class relations (types) versus subclass relations. Additional techniques can be applied to identify which surface strings denote entities and which ones refer to classes. Simple heuristics can already go a long way: for example, words that start with an uppercase letter are often entities whereas common nouns in plural form are more likely class names. Full-fledged approaches make use of dictionaries as discussed above or more advanced methods for entity recognition, discussed further below.

Hand-crafted patterns are useful also for discovering entities in semi-structured web contents like lists and tables. For example, if a list heading or a column header denotes a type, then the list or column entries could be considered as entities of that type. The pattern for this purpose would refer to HTML tags that mark headers and entries. Of course, this is just a crude heuristics that has a non-negligible risk of failing. We will discuss more advanced methods that handle this case more robustly Particularly, sections 6.2.1.5 and 6.3 go into depth on extraction from semi-structured contents for the more general scope of entity properties.

Multi-anchored Patterns:

Hearst patterns may pick up spurious observations. For example, the sentence "protest songs against war like Universal Soldier" could erroneously yield that `Universal Solider` is an instance of the class `wars`. One way of making the approach more robust is to include part-of-speech tags or even dependency-parsing trees (see [146, 273] for these NLP basics) in the specification of patterns. Another approach is to extend the context cue and strengthen its role. In addition to the pattern itself, we can demand that the context contains at least one additional entity which is already known to belong to the observed class. For example, the text "singers such as Elvis Presley" alone may be considered insufficient evidence to accept Elvis as a singer, but the text "singers such as Elvis Presley and Frank Sinatra" would have a stronger cue if `Frank Sinatra` is a known singer already. This idea has been referred to as *doubly-anchored patterns* in the literature [292] for the case of observing two entities of the same class. With a strong cue like "singers such as", insisting on a known witness may be an overkill, but the principle equally applies to weaker cues, for example, "voices such as ..." for the target class `singers`.

Multi-anchored patterns are particularly useful when going beyond text-based Hearst patterns by considering strong co-occurrence in enumerations, lists and tables. For simplicity, consider only the case of tables in web pages – as opposed to relational tables in databases. The co-occurring entities are typically the names in the cells of the same column, and the class is the name in the column header. Due the ambiguity of words and the ad-hoc nature of web tables, the spotted entities in the same column may be very heterogeneous, mixing up apples and oranges. For example, a table column on Oscar winners could have both actors and movies as rows. Thus, we may incorrectly learn that `Godfather` is an actor and `Bob Dylan` is a movie. To overcome these difficulties, we can require that a new entity name is accepted only when co-occurring with a certain number of known entities that belong to the proper class ([109]), for example, at least 10 actors in the same column for a table of 15 rows. Needless to say, all these are still heuristics that may occasionally fail, but they are easy to implement, powerful and have practical value.

4.3.2 Pattern Learning

Pre-specified patterns are inherently limited in their coverage. This motivates approaches for automatically learning patterns, using initial entity-type pairs and/or initial patterns for *distant supervision*. For example, when frequently observing a phrase like "great voice in" for entities of type `singers`, this phrase could be added to a set of indicative patterns for discovering more singers. This idea is captured in the following principle of *statement-pattern duality*, first formulated by [58] (see also [478] in the context of question answering):

Principle of Statement-Pattern Duality

When correct statements about entities x (e.g., x belonging to class y) frequently co-occur with textual pattern p, then p is likely a good pattern to derive statements of this kind.

Conversely, when statements about entities x frequently co-occur with a good pattern p, then these statements are likely correct.

Thus, observations of good statements and good patterns reinforce each other; hence the name statement-pattern duality.

This insightful paradigm gives rise to a straightforward algorithm where we start with statements in the KB as *seeds* (and possibly also with pre-specified patterns), and then iterate between deriving patterns from statements and deriving statements from patterns.

Algorithm for Seed-based Pattern Learning
Input: Seed statements in the form of known entities for a class
Output: Patterns for this class, and new entities of the class

Initialize:
$S \leftarrow$ seed statements
$P \leftarrow \emptyset$ (or pre-specified patterns like Hearst patterns)
Repeat

1. Pattern discovery:
   - search for mentions of entity $x \in S$ in web corpus, and identify co-occurring phrases;
   - generalize phrases into patterns by substituting x with a placeholder $\$X$;
   - analyze frequencies of patterns (and other statistics);
   - $P \leftarrow P \cup$ frequent patterns;
2. Statement expansion:
   - search for occurrences of patterns $p \in P$ in web corpus, and identify co-occurring entities;
   - analyze frequencies of entities co-occurring with multiple patterns (and other statistics);
   - $S \leftarrow S \cup$ frequent entities;

A toy example for running this algorithm is shown in Table 4.1.

In the example, phrases such as "$\$X$'s vocal performance" (with $\$X$ as a placeholder for entities) are generalized patterns. They co-occur with at least one but typically multiple of the entities in S known so far, and their strength is the cumulative frequency of these occurrences. Newly discovered patterns also co-occur with seed entities, and this would further strengthen the patterns' usefulness. The NELL project [394] has run a variant of this algorithm at large scale, and has found patterns for musicians, such as "original song by X", "ballads reminiscent of X", "bluesmen , including X", "was later covered by X", "also performed with X", "X 's backing bands", and hundreds more (see http://rtw.ml.cmu.edu/rtw/kbbrowser/predmeta:musician).

Despite its elegance, the algorithm, in this basic form, has severe limitations:

1. Over-specific patterns:
   Some patterns are overly specific. For example, a possible pattern "$\$X$ and her chansons" would apply only to female singers. This can be overcome by generalizing patterns into *regular expressions* over words and part-of-speech tags [153]: "$\$X$ $*$ and PRP chansons" in this example, where $*$ is a wildcard for any word sequence and PRP requires a personal pronoun. Likewise, the pattern "$\$X$'s great voice" could be generalized into "$\$X$'s JJ voice" to allow for other adjectives (with part-of-speech tag JJ) such as "haunting voice"

0	sentence:	Singers like Elvis Presley were kings of rock'n'roll.
1.1	new pattern:	*singers like* $X
1.2	sentence: new entity:	Singers like Nina Simone fused jazz, soul and gospel. `Nina Simone`
2.1	sentence: new pattern:	Nina Simone's vocal performance was amazing. $X *'s vocal performance*

2.2	sentence:
	new entity:
	sentence:
	new entity:
		Amy Winehouse's vocal performance won a Grammy.
		`Amy Winehouse`
		Queen's vocal performance led by Freddie Mercury ...
		`Queen`

3.1	sentence: sentence: new pattern:	The voice of Amy Winehouse reflected her tragic life. The voice of Elvis Presley is an incredible baritone. *voice of* $X

3.2	sentence:
	new entity:
	sentence:
	new entity:
		The melancholic voice of Francoise Hardy ...
		`Francoise Hardy`
		The great voice of Donald Trump got loud and angry.
		`Donald Trump`

...	...	...

Table 4.1: Toy example for Seed-based Pattern Learning, with seed "Elvis Presley"

or "angry voice". Moreover, instead of considering these as sequences over the surface text, the patterns could also be derived from paths in dependency-parsing trees (see, e.g., [65, 402, 561]).

2. Open-ended iterations:
   In principle, the loop could be run forever. Recall will continue to increase, but precision will degrade with more iterations, as the acquired patterns get diluted. So we need to define a meaningful stopping criterion. A simple heuristic could be to consider the fraction of seed occurrences obtained at the end of iteration i, as the new patterns should still co-occur with known entities. So a sudden drop in the observations of seeds would indicate a notable loss of quality.
3. False positives and pattern dilution:
   Even after one or two iterations, some of the newly observed statements are false positives: Queen is a band, not a singer, and Donald Trump does not have a lot of musical talent. This is caused by picking up overly broad or ambiguous patterns. For example, "Grammy winner" applies to bands as well, and "$X's great voice" may be observed in sarcastic news about politics, besides music.

As already stated for points 1 and 2, these weaknesses can be ameliorated by extending the method. The hardest issue is point 3. To mitigate the potential dilution of patterns, a number of techniques have been explored. One is to impose additional constraints for pruning out misleading patterns and spurious statements; we will discuss these in Chapter 6 for the more general task of acquiring relational statements. A second major technique is to compute statistical measures of pattern and statement quality after each iteration, and use these to drop doubtful candidates [3]. In the following, we list some of the salient measures that can be leveraged for this purpose.

The **support** of pattern p for seed statements S_0, $supp(p, S_0)$, is the ratio of the frequency of joint occurrences of p with any of the entities $x \in S_0$ to the co-occurrence frequency of any pattern with any $x \in S_0$:

$$supp(p, S_0) = \frac{\sum_{x \in S_0} freq(p, x)}{\sum_q \sum_{x \in S_0} freq(q, x)}$$

where $\sum_q$ ranges over all observed patterns q (possibly with a lower bound on absolute frequency) and $freq(\;)$ is the total number of observations of a pattern or pattern-entity pair.

The **confidence** of pattern p with regard to seed statements S_0 is the ratio of the frequency of p jointly with seed entities $x \in S_0$ to the frequency of p with any entities:

$$conf(p, S_0) = \frac{freq(p, S_0)}{freq(p)}$$

The **diversity** of pattern p with regard to seed statements S_0, $div(p, S_0)$, is the number of distinct entities $x \in S_0$ that co-occur with pattern p:

$$div(p, S_0) = |\{x \in S_0 \mid freq(p, x) > 0\}|$$

We can also contrast the positive occurrences of a pattern p, that is, co-occurrences with correct statements from S_0, against the negative occurrences with statements known to be incorrect. To this end, we have to additionally compile a set of incorrect statements as **negative seeds**, for example, specifying that the Beatles and Barack Obama are not singers to prevent noisy patterns that led to acquiring Queen and Donald Trump as new singers. Let us denote these negative seeds as $\overline{S}_0$. This allows us to revise the definition of confidence:

Given positive seeds S_0 and negative seeds $\overline{S}_0$, the **confidence** of pattern p, $\mathit{conf}(p)$, is the ratio of positive occurrences to occurrences with either positive or negative seeds:

$$\mathit{conf}(p) = \frac{\sum_{x \in S_0} \mathit{freq}(p, x)}{\sum_{x \in S_0} \mathit{freq}(p, x) \; + \; \sum_{x \in \overline{S}_0} \mathit{freq}(p, x)}$$

These quality measures can be used to restrict the acquired patterns to those for which support, confidence or diversity – or any combination of these – are above a given threshold. Moreover, the measures can also be carried over to the observed statements. This is again based on the principle of statement-pattern duality. To this end, we now identify a subset P of *good patterns* using the statistical quality measures.

The **confidence of statement** x (i.e., that an entity x belongs to the class of interest) is the normalized aggregate frequency of co-occurring with good patterns, weighted by the confidence of these patterns:

$$\mathit{conf}(x) = \frac{\sum_{p \in P} \mathit{freq}(x, p) \cdot \mathit{conf}(p)}{\sum_{q} \mathit{freq}(x, q)}$$

where $\sum_q$ ranges over all observed patterns. That is, we achieve perfect confidence in statement x if it is observed only in conjunction with good patterns and all these patterns have perfect confidence 1.0.

The **diversity of statement** x is the number of distinct patterns $p \in P$ that x co-occurs with:

$$\mathit{div}(x) = |\{p \in P \mid \mathit{freq}(p, x) > 0\}|$$

For both of these measures, variations are possible as well as combined measures. Diversity is a useful signal to avoid that a single pattern drives the acquisition of statements, which would incur a high risk of error propagation. Rather than relying directly on these measures, it is also possible to use probabilistic models or random-walk techniques for scoring and ranking newly acquired statements. In particular, algorithms for **set expansion** (aka. concept expansion) can be considered to this end (e.g., [612, 223, 608, 87]).

We can now leverage these considerations to extend the seed-based pattern learning algorithm. The key idea is to prune, in each round of the iterative method, both patterns and statements that do not exceed certain thresholds for support, confidence and/or diversity. Conversely, we can *promote*, in each round, the best statements to the status of seeds, to incorporate them into the calculation of the quality statistics for the next round. For example, when observing `Amy Winehouse` as a high-confidence statement in some round, we can add her to the seed set, this way enhancing the informativeness of the statistics in the next round. We sketch this extended algorithm, whose key ideas have been developed by [3] and [153].

Extended algorithm for Seed-based Pattern Learning
Input: Seed statements in the form of known entities for a class
Output: Patterns for this class, and new entities of the class

Initialize:
$S \leftarrow$ seed statements
$S^+ \leftarrow S$ //acquired statements
$P \leftarrow \emptyset$ (or pre-specified patterns like Hearst patterns)
Repeat

1. Pattern discovery:
   - same steps as in base algorithm
   - $P \leftarrow P \setminus$ {patterns below quality thresholds}
2. Statement expansion:
   - same steps as in base algorithm
   - $S^+ \leftarrow S^+ \cup$ {newly acquired statements}
   - $S \leftarrow S \cup$ {new statements above quality thresholds}

All of the above assumes that patterns are either matched or not. However, it is often the case that a pattern is almost but not exactly matched, with small variations in the wording or using synonymous words. For example, the pattern "Nobel prize winner $\$X$" (for `scientists` for a change) could be considered as *approximately matched* by phrases such as "Nobel winning $\$X$ or "Nobel laureate $\$X$". If these phrases are frequent and we want to consider them as evidence, we can add a *similarity kernel* $sim(p, q)$ to the observation statistics, based on *edit distance* or *n-gram overlap* where n-grams are sub-sequences of n consecutive characters. This simply extends the frequency of pattern p into a weighted count $freq(p) = \sum_q sim(p, q)$ if $sim(p, q) > \theta$ where $\sum_q$ ranges over all approximate matches of p in the corpus and θ is a pruning threshold to eliminate weakly matching phrases.

The presented techniques are also applicable to acquiring subclass/superclass pairs (for the `subclass-of` relation, as opposed to `instance-of`). For example, we can detect from patterns in web pages that `rappers` and `crooners` are subclasses of `singers`. This has been further elaborated in work on ontology/taxonomy learning like [363] and [153].

4.4 Machine Learning for Sequence Labeling

A major alternative to learning patterns for entity discovery is to devise **end-to-end machine learning** models. In contrast to the paradigm of seed-based distant supervision of Subsection 4.3, we now consider *fully supervised* methods that require **labeled training data** in the form of *annotated sentences* (or other text snippets). Typically, these methods work well only if the training data has a substantial size, in the order of ten thousand or higher. By exploiting large corpora with appropriate markup, or annotations from crowdsourcing workers, or high-quality outputs of employing methods like those for premium sources, such large-scale training data is indeed available today. On the first direction, Wikipedia is again a first-choice asset, as it has many sentences where a named entity appears and is marked up as a hyperlink to the entity's Wikipedia article.

In the following, we present two major families of end-to-end learning methods. The first one is **probabilistic graphical models** where a sequence of words, or, more generally, *tokens*, is mapped into the joint state of a graph of random variables, with states denoting tags for the input words. The second approach is **deep neural networks** for classifying the individual words of an input sequence onto a set of tags. Thus, both of these methods are geared for the task of **sequence labeling**, also known as **sequence tagging**.

4.4.1 Probabilistic Graphical Models

This family of models considers a set of coupled random variables that take a finite set of tags as values. In our application, the tags are primarily used to demarcate entity names in a token sequence, like an input sentence or other snippet. Each random variable corresponds to one token in the input sequence, and the coupling reflects short-distance dependencies (e.g., between the tags for adjacent or nearby tokens). In the simplest and most widely used case, the coupling is pair-wise such that a random variable for an input token depends only on the random variable for the immediately preceding token. This setup amounts to learning conditional probabilities for subsequent pairs of tags as a function of the input tokens. As the input sequence is completely known upfront, we can generalize this to each random variable being a function of all tokens or all kinds of *feature functions* over the entire token sequence.

Conditional Random Fields (CRF):
The most successful method from this family of probabilistic graphical models is known as **Conditional Random Fields**, or **CRF**s for short, originally developed by [304]. More recent tutorials are by [569] on foundations and algorithms, and [511] on applying CRFs for information extraction. CRFs are in turn a generalization of the prior notion of *Hidden Markov Models (HMMs)*, the difference lying in the incorporation of feature functions on the entire input sequence versus only considering two successive tokens.

A **Conditional Random Field (CRF)**, operating over input sequence $X = x_1 \ldots x_n$ is an undirected graph, with a set of finite-state random variables $Y = \{Y_1 \ldots Y_m\}$ as nodes and pair-wise coupling of variables as edges.
An edge between variables Y_i and Y_j denotes that their value distributions are coupled. Conversely, it denotes that, in the absence of any other edges, two variables are conditionally independent, given their neighbors.
More precisely, the following *Markov condition* is postulated for all variables Y_i and all possible values t:

$$P[Y_i = t \mid x_1 \ldots x_n,\ Y_1 \ldots Y_{i-1}, Y_{i+1} \ldots Y_m] =$$

$$P[Y_i = t \mid x_1 \ldots x_n,\ \text{all } Y_j \text{ with an edge } \ (Y_i, Y_j)]$$

Strictly speaking, the coupling of random variables may go beyond pairs by introducing *factor nodes* (or *factors* for short) for dependencies between two or more variables. In this section, we restrict ourselves to the basic case of pair-wise coupling. Moreover, we assume that the graph forms a linear chain: a so-called **linear-chain CRF**. Often the variables correspond one-to-one to the input tokens: so each token x_i is associated with variable Y_i (and we have $n = m$ for the number of tokens and variables).

CRF Training:
The *training* of a CRF from labeled sequences, in the form of (X, Y) value pairs, involves the posterior likelihoods

$$P[Y_i|X] = P[Y_i = t_i \mid x_1 \ldots x_n,\ \text{all neighbors } Y_j \text{ of } Y_j]$$

With feature functions f_k over input X and subsets $Y_c \subset Y$ of coupled random variables (with known values in the training data), this can be shown to be equivalent to

$$P[Y|X] \sim \frac{1}{Z} \prod_c \exp(\sum_k w_k \cdot f_k(X,\ Y_c)]$$

with an input-independent normalization constant Z, k ranging over all feature functions, and c ranging over all coupled subsets of variables, the so-called *factors*. For a linear-chain CRF, the factors are all pairs of adjacent variables:

$$P[Y|X] \sim \frac{1}{Z} \prod_i \exp(\sum_k w_k \cdot f_k(X, Y_{i-1}, Y_i)]$$

The parameters of the model are the feature-function weights w_k; these are the output of the training procedure. The training objective is to choose weights w_k to minimize the error between the model's maximum-posterior Y values and the ground-truth values, aggregated over all training samples. The error, or *loss function*, can take different forms, for example, the negative log-likelihood that the trained model generates the ground-truth tags.

As with all machine-learning models, the objective function is typically combined with a *regularizer* to counter the risk of overfitting to the training data. The training procedure is usually implemented as a form of *(stochastic) gradient descent* (see, e.g., [68] and further references given there). This guarantees convergence to a local optimum and empirically approximates the global optimum of the objective function fairly well (see [569]). For the case of linear-chain CRFs, the optimization is convex; so we can always approximately achieve the global optimum.

CRF Inference:
When a trained CRF is presented with a previously unseen sentence, the *inference* stage computes the posterior values of all random variables, that is, the tag sequence for the entire input, that has the *maximum likelihood* given the input and the trained model with weights w_k:

$$Y^* = \underset{Y}{\operatorname{argmax}}\, P[Y \mid X,\ \text{all } w_k]$$

For linear-chain CRFs, this can be computed using *dynamic programming*, namely, variants of the Viterbi algorithm (for HMMs). For general CRFs, other – more expensive – techniques like *Monte Carlo sampling* or variational inference are needed. Alternatively, the CRF inference can also be cast into *an Integer Linear Program (ILP)* and solved by optimizers like the Gurobi software (https://www.gurobi.com/), following [500]. We will go into more depth on ILPs as a modeling and inference tool in Chapter 8, especially Section 8.5.3.

CRF for Part-of-Speech Tagging:
A classical application of CRF-based learning is part-of-speech tagging: labeling each word in an input sentence with its word category, like noun (NN), verb (VB), preposition (IN), article (DET) etc. Figure 4.1 shows two examples for this task, with their correct output tags.

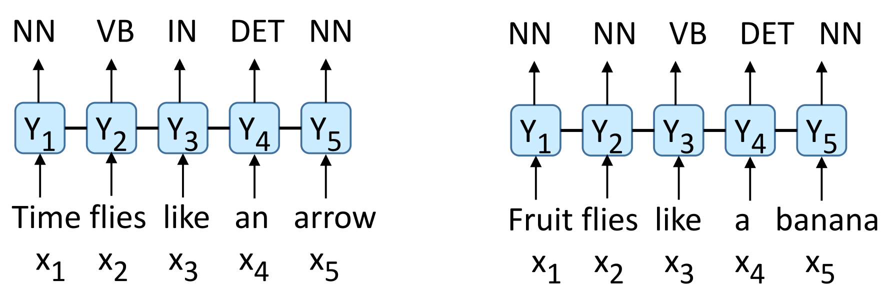


Figure 4.1: Examples for CRF-based Part-of-Speech Tagging

The intuition why this works so well is that word sequence frequencies as well as tag sequence frequencies from large corpora can inform the learner, in combination with other feature functions, to derive very good weights. For example, nouns are frequently followed by verbs, verbs are frequently followed by prepositions, and some word pairs are composite noun phrases (such as "fruit flies"). Large corpus statistics also help to cope with exotic or even non-sensical inputs. For example, the sentence "Pizza flies like an eagle" would be properly tagged as *NN VB IN DET NN* because, unlike fruit flies, there is virtually no mention of pizza flies in any corpus.

CRF for Named Entity Recognition (NER) and Typing:
For the task at hand, entity discovery, the CRF tags of interest are primarily NE for Named Entity and O for Others. For example, the sentence

"Dylan composed the song Sad-eyed Lady of the Lowlands,
about his wife Sara Lownds, while staying at the Chelsea hotel"

should yield the tag sequence

NE O O O NE NE NE NE NE O O O NE NE O O O O NE NE.

Many entity mentions correspond to the part-of-speech tag NNP, for proper noun, that is, nouns that should not be prefixed with an article in any sentence, such as names of people. However, this is not sufficient, as it would accept false positives like abstractions (e.g., "love" or "peace") and would miss out on multi-word names that include non-nouns, such as song or book titles (e.g. "Sad-eyed Lady of the Lowlands"), and names that come with an article (e.g., "the Chelsea hotel"). For these reasons, CRFs for *Named Entity Recognition*, or *NER* for short, have been specifically developed, with training over annotated corpora. The seminal work on this is [166]; advanced extensions are implemented in the Stanford CoreNLP software suite ([367]).

As the training of a CRF involves annotated corpora, instead of merely distinguishing entity mentions versus other words, we can piggyback on the annotation effort and incorporate more expressive tags for *different types of entities*, such as people, places, products (incl. songs and books). This idea has indeed been pursued already in the work of [166], integrating **coarse-grained entity typing** into the CRF for NER. The simple change is to move from output variables with tag set $\{NE, O\}$ to a larger tag set like $\{PERS, LOC, ORG, MISC, O\}$ denoting persons (PERS), locations (LOC), organizations (ORG), entities of miscellanous types (MISC) such as products or events, and non-entity words (O). For the above example about Bob Dylan, we should then obtain the tag sequence

PERS O O O MISC MISC MISC MISC MISC
O O O PERS PERS O O O O LOC LOC.

The way the CRF is trained and used for inference stays the same, but the training data requires annotations for the entity types. Later work has even devised (non-CRF) classifiers for more **fine-grained entity typing**, with tags for hundreds of types, such as politicians, scientists, artists, musicians, singers, guitarists, etc. [168, 344, 413, 93]. An easy way of obtaining training data for this task is consider hyperlink anchor texts in Wikipedia as entity names and derive their types from Wikipedia categories, or directly from a core KB constructed by methods from Chapter 3. An empirical comparison of various NER methods with fine-grained typing is given by [365].

Widely used feature functions for CRF-based NER tagging, or features for other kinds of NER/type classifiers, include the following:

- part-of-speech tags of words and their co-occurring words in left-hand and right-hand proximity,
- uppercase versus lowercase spelling,
- word occurrence statistics in type-specific dictionaries, such as dictionaries of people names, organization names, or location names, along with short descriptions (from yellow pages and so-called gazetteers),
- co-occurrence frequencies of word-tag pairs in the training data,
- further statistics for word n-grams and their co-occurrences with tags.

There is also substantial work on **domain-specific NER**, especially for the biomedical domain (see, e.g., [170] and references given there), and also for chemistry, restaurant names and menu items, and titles of entertainment products. In these settings, domain-specific dictionaries play a strong role as input for feature functions [476, 519].

4.4.2 Deep Neural Networks

In recent years, neural networks have become the most powerful methodology for supervised machine learning when sufficient training data are available. This holds for a variety of NLP tasks [186], potentially including Named Entity Recognition.

The most primitive neural network is a single *perceptron* which takes as input a set of real numbers, aggregates them by weighted summation, and applies a *non-linear activation function* (e.g., logistic function or hyperbolic tangent) to the sum for producing its output. Such building blocks can be connected to construct entire networks, typically organized into layers. Networks with many layers are called *deep networks*. As a loose metaphor, one may think of the nodes as neurons and the interconnecting edges as synapses.

The weights of incoming edges (for the weighted summation) are the parameters of such neural models, to be learned from labeled training data. The inputs to each node are usually entire vectors, not just single numbers, and the top-layer's output are real values for regression models or, after applying a softmax function, scores for classification labels. The loss function for the training objective can take various forms of error measures, possibly combined with regularizers or constraint-based penalty terms. For neural learning, it is crucial that the loss function is differentiable in the model parameters (i.e., the weights), and that this can be backpropagated through the entire network. Under this condition, training is effectively performed by methods for *stochastic gradient descent* (see, e.g., [68]), and modern software libraries (e.g., TensorFlow) support scaling out these computations across many processors. For inference, with new inputs outside the training data, input vectors are simply fed forward through the network by performing matrix and tensor operations at each layer.

LSTM Models:
There are various families of neural networks, with different topologies for interconnecting layers. For NLP where the input to the entire network is a text sequence, so-called **LSTM networks** (for "Long Short Term Memory") have become prevalent (see [515, 186] and references there). They belong to the broader family of *recurrent neural networks* with feedback connections between nodes of the same layer. This allows these nodes to aggregate latent state computed from seeing an input token and the latent state derived from the preceding tokens. To counter potential bias from processing the input sequence in forward direction alone, **Bi-directional LSTM**s (or **Bi-LSTMs** for short) connect nodes in both directions.

We can think of LSTMs as the neural counterpart of CRFs. A key difference, however, is that neural networks do not require the explicit modeling of feature functions. Instead, they take the raw data (in vectorized form) as inputs and automatically learn latent representations that implicitly capture features and their cross-talk.

Figure 4.2 gives a pictorial illustration of an LSTM-based neural network for NER, applied to a variant of our Bob Dylan example sentence. The outputs of the forward LSTM and the backward LSTM are combined into a latent data representation, for example, by concatenating vectors. On top of the bi-LSTM, further network layers (learn to) compute the scores for each tag, typically followed by a softmax function to choose the best label. The output sequence of tags is slightly varied here, by prefixing each tag with its role in a subsequence of identical tags: B for Begin, I for In, and E for End. This serves to distinguish the case of a single multi-word mention from the case of different mentions without any interleaving "Other" words. The E tags are not really needed as a following B tag indicates the next mention anyway, hence E is usually omitted. This simple extension of the tag set is also adopted by CRFs and other sequence labeling learners; we disregarded this earlier for simplicity.

LSTM-based neural networks can be combined with a CRF on top of the neural layers [250, 361, 306], this way combining the strengths of the two paradigms. Other enhancements (see [330] for a survey of neural NER) include additional bi-LSTM layers for learning character-level representations, capturing character n-grams in a latent manner. This can leverage large unlabeled corpora, analogously to the role of dictionaries in feature-based taggers. This line of methods has also extended the scope of *fine-grained entity typing*, yielding labels for thousands of types [93].

Overall, deep neural networks, in combination with CRFs, tend to outperform other methods for NER whenever a large amount of training data is at hand. When training data

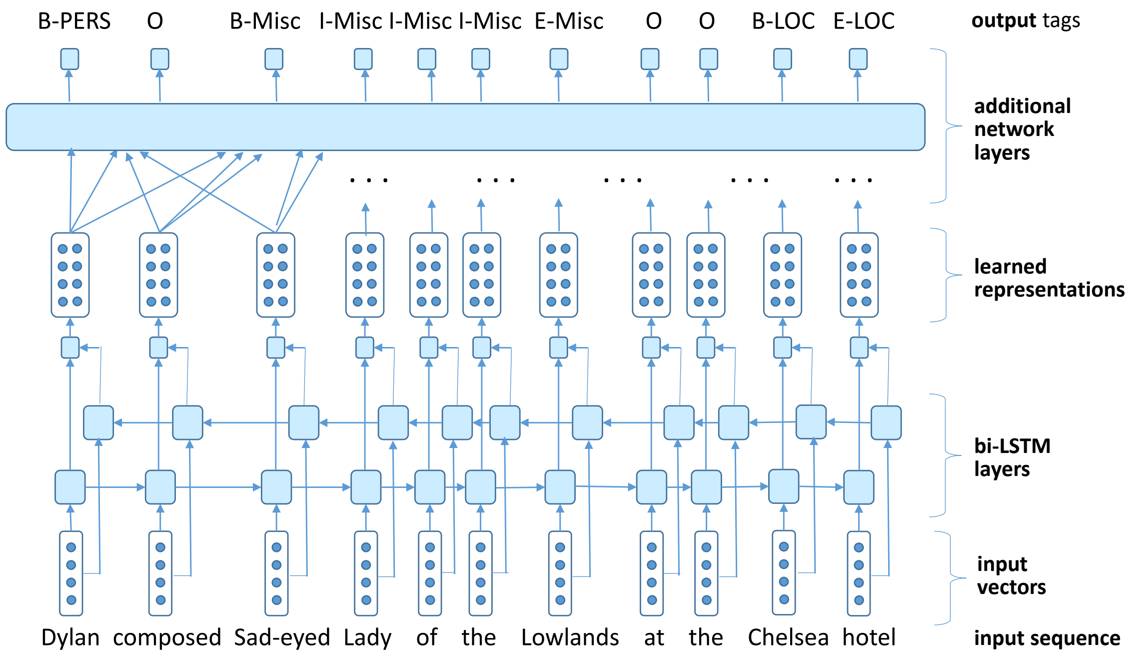


Figure 4.2: Illustration of LSTM network for NER

is not abundant, for example, in specific domains such as health, pattern-based methods and feature-driven graphical models (incl. CRFs) are still a good choice.

4.5 Word and Entity Embeddings

Machine learning methods, and especially neural networks, do not operate directly on text, as they require numeric vectors as inputs. A popular way of casting text into this suitable form is by means of **embeddings**.

> **Word Embedding:**
> Embeddings of words (or multi-word phrases) are real-valued vectors of fixed dimensionality, such that the distance between two vectors (e.g., by their cosine) reflects the *relatedness* (sometimes called "semantic similarity") of the two words, based on the respective contexts in which the words typically occur.

Word embeddings are computed (or "learned") from co-occurrences and neighborhoods of words in large corpora. The rationale is that the meaning of a word is captured by the contexts in which it is often used, and that two words are highly related if they are used in similar contexts. This is often referred to as "distributional semantics" or "distributed semantics" of words [324].

This hypothesis should not be confused with two words directly co-occurring together. Instead, we are interested in indirect co-occurrences where the contexts of two input words share many words. For example, the words "car" and "automobile" have the same semantics, but rarely co-occur directly in the same text span. The point rather is that both often co-occur with third words such as "road", "traffic", "highway" and names of car models.

Technically, this becomes an optimization problem. Given a large set of text windows C of length $k+1$ with word sequences $w_0 \dots w_k$, we aim to compute a fixed-length vector $\vec{w}$ for each word w such that the error for predicting the word's surrounding window $C(w_t) = w_{t-k/2} \dots w_t \dots w_{t+k/2}$, from all word-wise vectors alone, is minimized. This consideration leads to a non-convex continuous optimization with the objective function [384]:

$$maximize \quad \sum_{C} \sum_{j \in C(w_t), j \neq t} \log \frac{exp(\vec{w_j}^T \cdot \vec{w_t})}{\sum_v exp(\vec{v}^T \cdot \vec{w_t})}$$

where the outermost sum ranges over all possible text windows (with overlapping windows). The dot product between the output vectors $\vec{w_j}$ and $\vec{w_t}$ reflects overlapping-context likelihoods of word pairs, and the softmax function normalizes these scores by considering all possible words v. Intuitively, the objective is maximized if the resulting word vectors can predict the surrounding window from a given word with high accuracy. This specific objective is known as the *skip-gram model*; there are also other variations, with a similar flavor. Computing solutions for the non-convex optimization is typically done via gradient descent methods.

Embedding vectors can be readily plugged into machine-learning models, and they are a major asset for the power of neural networks for NLP tasks. A degree of freedom is the choice for the dimensionality of the vectors. For most use cases, this is set to a few hundred, say 300, for robust behavior.

The most popular models and tools for this line of text embeddings are word2vec, by [384], and GloVE, by [447]. Both come with pre-packaged embeddings derived from news and other collections, but applications can also compute new embeddings from customized corpora. The word2vec approach has been further extended to compute embeddings for short paragraphs and entire documents (called doc2vec). Important earlier work on latent text representations, with similar but less expressive models, includes Latent Semantic Indexing (LSI) and Latent Dirichlet Allocation (LDA) ([114, 243, 48]).

A recent, even more advanced way of computing and representing embeddings is by training deep neural networks for word-level or sentence-level prediction tasks, and then keeping the learned model as a building block for training an encompassing network for downstream tasks (e.g., question answering or conversational chatbots).

The pre-training utilizes large corpora like the full text of all Wikipedia articles or the Google Books collection. A typical objective function is to minimize the error in predicting a masked-out word given a text window of successive words. The embeddings for all words are jointly given by the learned weights of the network's synapses (i.e., connections between neurons), with 100 millions of real numbers or even more. Popular instantiations of this approach are ElMo [448] and BERT [118].

Embeddings for Word and Entity Pair Relatedness:
Embedding vectors are not directly interpretable; they are just vectors of numbers. However, we can apply linear algebra operators to them to obtain further results. Embeddings are additive and subtractive, which allows forming analogies of the form:

$$\begin{aligned}
\overrightarrow{man} + \overrightarrow{king} &= \overrightarrow{woman} + \overrightarrow{queen} \\
\overrightarrow{France} + \overrightarrow{Paris} &= \overrightarrow{Germany} + \overrightarrow{Berlin} \\
\overrightarrow{Einstein} + \overrightarrow{scientist} &= \overrightarrow{Messi} + \overrightarrow{footballer}
\end{aligned}$$

So we can solve an equation like

$$\overrightarrow{Rock\ and\ Roll} + \overrightarrow{Elvis\ Presley} = \overrightarrow{Folk\ Rock} + \overrightarrow{X}$$

yielding

$$\overrightarrow{X} = \overrightarrow{Rock\ and\ Roll} + \overrightarrow{Elvis\ Presley} - \overrightarrow{Folk\ Rock} \quad \approx \overrightarrow{Bob\ Dylan}$$

Most importantly for practical purposes, we can compare the embeddings of two words (or phrases) by computing a distance measure between their respective vectors, typically, the cosine or the scalar product. This gives us a measure of how strongly the two words are related to each other, where (near-) synonyms would often have the highest relatedness.

Embedding-based Relatedness:
For two words v and w, their relatedness can be computed as $cos(\vec{v}, \vec{w})$ from their embedding vectors $\vec{v}$ and $\vec{w}$.

The absolute values of the relatedness scores are not crucial, but we can now easily order related words by descending scores. For example, for the word "rock", the most related words and short phrases are "rock n roll", "band", "indie rock" etc., and for "knowledge" we obtain the most salient words "expertise", "understanding", "knowhow", "wisdom" etc. We will later see that such relatedness measures are very useful for many sub-tasks in knowledge base construction.

The embedding model can capture not just words or other text spans, but we can also apply it to compute **distributional representations of entities**. This is achieved by associating each entity in the knowledge base with a textual description of the entity, typically the Wikipedia article about the entity (but possibly also the external references given there, homepages of people and organizations, etc.).

Once we have per-entity vectors, we can again compute cosine or scalar-product distances for entity pairs. This results in measures for **entity-entity relatedness**. Moreover, by coupling the computations of per-word and per-entity embeddings, we also obtain scores for **entity-word relatedness**, which is often handy when we need salient keywords or keyphrases for an entity. For example, the embedding for Elvis Presley should be close to the embeddings for “king”, “rock n roll”, etc. Technical details for these models can be found in [616, 677, 638]; the latter includes data and code for the *wikipedia2vec* tool.

Such embeddings have also been computed from domain-specific data sources, most notably, for biomedical entities and terminology, with consideration of the standard MeSH vocabulary. Resources of this kind include BioWordVec ([663]) and BioBERT ([314]).

An important predecessor to all these works is the semantic relatedness model of [171], which was the first to harness Wikipedia articles for this purpose.

A related, recent direction is *knowledge graph (KG) embeddings* (see [611] for a survey). These kinds of embeddings capture the neighborhood of entities in an existing graph-structured KB. They do not use textual inputs, however, and serve different purposes. We will discuss KG embeddings in Chapter 8, specifically Section 8.4.

4.6 Ab-Initio Taxonomy Construction

Assuming that we can extract a large pool of types, and optionally also entities for them, the task discussed here is to construct a taxonomic tree or DAG (directed acyclic graph) for these types – without assuming any prior structure such as WordNet. In the literature, the problem is also referred to as **taxonomy induction** [541, 457], as its output is a generalization of bottom-up observations. The input can take different forms, for example, starting from the noisy set of Wikipedia categories (but ignoring the graph structure), or from noisy and sparse pairs of hyponym-hypernym candidates derived by applying patters to large text and web corpora. A good example for the latter is the *WebIsALOD* project (http://webisa.webdatacommons.org/, which used more than 50 patterns to extract candidate pairs and a supervised classifier to prune out the most noisy ones [232]. This collection, and others of similar flavor, does not strictly focus on hypernymy but also captures meronymy/holonymy (part-of) and, to some extent, instance-type pairs. Hence the broader term *IsA* in the project name.

Methods for Wikipedia Categories:
Seminal work that considered all Wikipedia categories as noisy type candidates and the subcategory-supercategory pairs as hypernymy candidates was the *WikiTaxonomy* project [456, 457]. Its approach can be characterized by three steps:

Wikipedia-based Taxonomy Induction:

- *Category Cleaning:* eliminating noisy categories that do not really denote types.
- *Category-Pair Classification:* using a rule-based classifier to eliminate pairs that do not denote hypernymy.
- *Taxonomy Graph Construction:* building a tree or DAG from the remaining types and hypernymy pairs.

The first step is very similar to the techniques presented in Section 3.2. State-of-the-art techniques for this purpose are discussed in [440]. The second step is based on heuristic but powerful rules that compare stems or lemmas of head words in multi-word noun phrases. The following shows two examples for rules:

- For sub-category S and direct super-category C: if $head(S)$ is the same as $head(C)$, then this is likely a hyponym-hypernym pair (e.g., S = "American baritones", C = "baritones by nationality").
- For C and S: if $head(C)$ appears in S, but $head(S)$ is different from $head(C)$, then this is likely not a good pair (e.g., S = "American baritones", C = "baritone saxophone players"),

Additional rules are used to refine the first case and to handle other cases. This includes considering instances of a category, at the entity level, and comparing their set of categories against the category at hand.

For the third step, *graph construction*, the method applies transitivity to build a multi-rooted graph, eliminates cycles by removing as few edges as possible, and connects all roots of the resulting DAG to the universal type `entity`.

An industrial-strength variation and extension of the presented method is discussed in [117]. Another Wikipedia component that has been considered as noisy input for taxonomy induction is *infobox templates*. The Wikipedia community has developed a large number of different templates for people, musicians, bands, songs, albums etc. They are instantiated in highly varying numbers, and there is redundancy, for example, different templates for songs, some used more than others. [625] proposed a learning approach, using SVM classifiers and CRF-like graphical models, to infer a clean taxonomy from this noisy data.

Taxonomies from Catalogs, Networks and User Behavior:
Alternatively to Wikipedia categories, other catalogs of categories can be processed in a similar manner, for example, the DMOZ directory of web sites (https://dmoz-odp.org/) or the Icecat open product catalog (https://icecat.biz/). Some methods combine information from catalogs with topical networks, for example, connecting business categories, users and reviews on sites such as Yelp or TripAdvisor, and potentially also informative terms from user reviews. Examples of such methods are [609, 520]. Last but not least, recent methods

on this task start with a high-quality product catalog and its category hierarchy, and then learn to extend and enrich the taxonomy with input from other sources, most notably, logs of customer queries, clicks, likes and purchases. This method is part of the **AutoKnow** pipeline [133], discussed further in Section 9.5.

Folksonomies from Social Tags:
The general approach has also been carried over to build taxonomies from *social tagging*, resulting in so-called **folksonomies** [199]. The input is a set of items like images or web pages of certain types that are associated with concise *tags* to annotate items, such as "sports car", "electric car", "hybrid auto" etc. If the number of items and the tagging community are very large, the frequencies and co-occurrences of (words in) tags provide cues about proper types as well as type pairs where one is subsumed by the other. Data mining techniques (related to association rules) can then be applied to clean such a large but noisy candidate pool, and the subsequent DAG construction is straightforward (e.g.,[233, 247, 262]).

Another target for similar techniques are fan communities (e.g., hosted at http://fandom.com aka. Wikia), which have collaboratively built extensive but noisy category and tagging systems for entertainment fiction like movie series or TV series (e.g., Lord of the Rings, Game of Thrones, The Simpsons etc.) [97, 231].

Methods for Web Contents:
Early approaches spotted entity names in web-page collections and clustered these by various similarity measures (e.g., [141]). Using Hearst patterns and other heuristics, type labels are then derived for each cluster. Such techniques can be further refined for scoring and ranking the outputs (e.g., using label propagation with random-walk techniques [571]).

The seminal **KnowItAll** project [153] advanced this line of research by a suite of scoring and classification techniques to enhance the output quality. It also studied tapping into *lists of named entities* as a source of type cues. For example, headers or captions of lists may serve as type candidates, and pairs of lists where one mostly subsumes the other in terms of elements (with some tolerance for exceptions) can be viewed as candidates for hypernymy.

In the **Probase** project, noisy candidates for hypernymy pairs were mined from the Web index of a major search engine [631]. First, Hearst patterns were liberally applied to this huge text collection. Then, a probabilistic model was used to prune out noise and infer likely candidates for hyponym-hypernym pairs, based on (co-)occurrence frequencies. The approach led to a huge but still noisy and incomplete taxonomy. Also, the resulting types are not canonicalized, meaning that synonymous type names may appear as different nodes in the taxonomy with different neighborhoods of hyponyms and hypernyms. Nevertheless, for use cases like query recommendation in web search, such a large collection of taxonomic information can be a valuable asset.

Recent works approached taxonomy induction as a supervised machine-learning task, using factors graphs or neural networks, or by reinforcement learning (see, e.g., [31, 529, 368]).

Methods for Query-Click Logs:
Search engine companies have huge logs of query-click pairs: user-issued keyword queries and subsequent clicks on web pages after seeing the preview snippets of top-ranked results. When a sufficiently large fraction of queries is about types (aka. classes), such as "American song writers" or "pop music singers from the midwest", one can derive various signals towards inferring type synonymy and pairs for the IsA relation:

- *Surface cues in query strings:* frequent patterns of query formulations that indicate type names. Examples are queries that start with "list of" or noun-phrase query strings with a prefix (or head word) known to be a type followed by a modifier, such as "musicians who were shot" or "IT companies started in garages".
- *Co-Clicks:* pages that are (frequently) clicked upon two different queries. For example, if the queries "American song writers" and "Americana composers" have many clicks in common, they could be viewed as synonymous types.
- *Overlap of query and page title:* the word-level n-gram overlap between the query string and the title of a (frequently) clicked page. For example, if the query "pop music singers from the midwest" often leads to clicking the page with title "baritone singers from the midwest", this pair is a candidate for the IsA relation (or, specifically, hypernymy between types if the two strings are classified to denote types, not entity instances or other phrases).

A variety of methods have been devised to harness these cues for inferring taxonomic relations (synonymy and hypernymy, or more coarsely IsA) by [24, 441, 347, 346]. These involve scoring and ranking the candidates, so that different slices can be compiled depending on whether the priority is precision or recall. By incorporating word-embedding-based similarities and learning techniques, the directly observed cues can also convey generalizations, for example, inferring that "crooners from Mississippi" are a subtype of "singers from the midwest".

Some of the resulting collections of types and type-name pairs are much richer and more fine-grained than the taxonomies that hinge on Wikipedia-like sources. Their strength is that they cover very specific types absent in current KBs, such as "musicians who were shot" (e.g., John Lennon), "musicians who died at 27" (e.g., Jim Morrison, Amy Winehouse, etc.), or "IT companies started in garages" (e.g., Apple), along with sets of paraphrases for these (e.g., "27 club" for "musicians who died at 27"). Such repositories are very useful for query suggestions (i.e., auto-completion or re-formulations) and explorative browsing (see, e.g., [347, 346]), but they do not (yet) reach the semantic rigor and near-human quality of full-fledged knowledge bases.

Discussion:
Overall, the methods presented in this section have not yet achieved taxonomies of better quality and much wider coverage than those built directly from premium sources (see Chapter 3). Nevertheless, the outlined methodologies for coping with noisier input are of interest and value, for example, for query suggestion by search engines and towards constructing domain-specific KBs (e.g. on health where user queries could be valuable cues; see [286] and references there).

4.7 Take-Home Lessons

The following are key points to remember.

- The task of entity discovery involves finding more entities for a given type as well as finding more informative types for a given entity. These two goals are intertwined, and many methods in this chapter apply to both of them.
- To discover entity names in web contents, *dictionaries* and *patterns* are an easy and very effective way. Patterns can be hand-crafted, such as Hearst patterns, or automatically computed by *seed-based distantly supervised learning*, following the principle of *statement-pattern duality*. For assuring the quality of newly acquired entity-type pairs, quantitative measures like *support* and *confidence* must be considered.
- When sufficient amounts of *labeled training data* are available, in the form of annotated sentences, *end-to-end supervised learning* is a powerful approach. This is typically cast into a sequence tagging task, known as *Named Entity Recognition (NER)* and *Named Entity Typing*. Methods for this purpose are based on probabilistic graphical models like *CRFs*, or deep neural networks like *LSTMs*, or combinations of both.
- A useful building block for all these methods are *word and entity embeddings*, which latently encode the degree of relatedness between pairs of words or entities.
- While most of these methods start with a core KB that already contains a (limited) set of entities and types, it is also possible to compute IsA relations by *ab-initio taxonomy induction* from text-based observations only. This has potential for obtaining more long-tail items, but comes at a higher risk of quality degradation.

5 Entity Canonicalization

5.1 Problem and Design Space

The entity discovery methods discussed in Chapter 4 may inflate the KB with alias names that refer to the same real-world entity. For example, we may end up with entity names such as "Elvis Presley", "Elvis" and "The King", or "Harry Potter Volume 1" and "Harry Potter and the Philosopher's Stone". If we treated all of them as distinct entities, we would end up with redundancy in the KB and, eventually, inconsistencies. For example, the birth and death dates for `Elvis Presley` and `Elvis` could be different, causing uncertainty about the correct dates. For some KB applications, this kind of inconsistency may not cause much harm, as long as humans are satisfied with the end results, such as finding songs for a search-engine query about `Elvis`. However, applications that *combine*, *compare* and *reason* with KB data, such as entity-centric analytics or recommendations, need to be aware of cases when two names denote the same entity. For example, counting book mentions for market studies should properly combine the two variants of the same Harry Potter book while avoiding conflation with other book titles that denote different volumes of the series. Likewise, a user should not erroneously get recommendations for a book that she already read.

This motivates why a high-quality KB needs to tame ambiguity by *canonicalizing* entity mentions, creating one entry for all observations of the same entity regardless of name variants. The task comes in a number of different settings.

5.1.1 Entity Linking (EL)

The most widely studied case is called **Entity Linking (EL)**, where we assume an existing KB with a rich set of canonicalized entities (e.g., harvested from premium sources) and we observe a new set of **mentions** in additional inputs like text documents or web tables. When the input is text, the task is also known as **Named Entity Disambiguation (NED)** in the computational linguistics community. Historically, the so-called *Wikification* task [383, 386] has aimed to map both named entities and general concepts onto Wikipedia articles (including common nouns such as "football", which could mean either American football or European football aka. soccer, or the ball itself). This leads to the broad task of **Word Sense Disambiguation (WSD)**. Text with entity mentions also contains general words that are often equally ambiguous. For example, words like "track" and "album" (cf. Figure 5.1) can have several and quite different meanings, referring to music (as in the figure) or to completely different topics. The WSD task is to map these surface words (and possibly also multi-word phrases) onto their proper word senses in WordNet or onto Wikipedia articles [417, 399, 203]. For the mission of KB construction, general concepts and WSD are out of scope.

Figure 5.1 gives an example for the EL task. In the input text on the left side, NER methods can detect mentions, and these need to be mapped to their proper entities in the KB, shown on the right-hand side. Candidate entities can be determined based on surface cues like string similarity of names. In the example, this leads to many candidates for the first name Bob, but also for the highly ambiguous mentions "Hurricane", "Carter" and "Washington". As each mention has so many mapping options, we face a complex combinatorial problem. Wikipedia knows more than 700 people with first name (or nickname) Bob, and Wikidata contains many more.

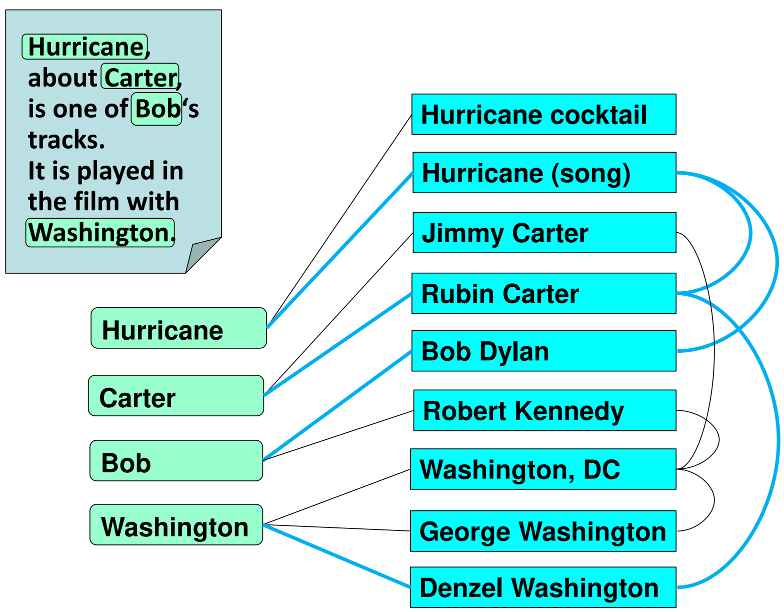


Figure 5.1: Example for Entity Linking

To compute, or learn to compute, the correct mapping, all methods consider various signals that relate input and output:

Mention-Entity Popularity:
If an entity is frequently referred by the name of the mention, this entity is a likely candidate. For example, "Carter" and "Washington" most likely denote the former US president `Jimmy Carter` and `Washington, DC`.

Mention-Entity Context Similarity:
Mentions have surrounding text, which can be compared to descriptions of entities such as short paragraphs from Wikipedia or keyphrases derived from such texts. For example, the context words "tracks" and "played" are cues towards music and musicians, and "film with" suggests that "Washington" is an actor or actress.

Entity-Entity Coherence:
In meaningful texts, different entities do not co-occur uniformly at random. Why would someone write a document about `Jimmy Carter` drinking a `Hurricane (cocktail)` together with `Robert Kennedy` and `George Washington`?
For two entities to co-occur, a semantic relationship should hold between them. The existing KB may have such prior knowledge that can be harnessed. For example, `Bob Dylan` has composed `Hurricane (song)`, and its lyrics is about the Afro-American boxer `Rubin Carter`, aka. Hurricane, who was wrongfully convincted for murder in the 1970s and later released after 20 years in prison. By mapping the mentions to these inter-related entities, we obtain a highly coherent interpretation.

In Figure 5.1, the edges between mentions and candidate entities indicate mention-entity similarities, and the edges among candidate entities indicate entity-entity coherence. Obviously, for quantifying the strength of similarity and coherence, these edges should be *weighted*, which is not shown in the figure. Some edge weights are stronger than others, and these are the cues for inferring the proper mapping. In Figure 5.1, these indicative edges are thicker lines in blue.
Algorithmic and learning-based methods for EL are based on these three components. EL is intensively researched and applied not just for KB construction, with comprehensive surveys by [523, 343, 370] and widely used benchmarks (e.g., [494]). This chapter discusses major families of methods and their building blocks.

5.1.2 Handling Out-of-KB Entities

The EL task has many variations and extensions. An important case is the treatment of **out-of-KB entities**: mentions that denote entities that are not (yet) included in the KB. This situation often arises with **emerging entities**, such as newly created songs or books, people or organizations that suddenly become prominent, and **long-tail entities** such as garage bands or small startups. In such cases, the EL method has an additional option to map a mention to *null*, meaning that none of the already known entities is a proper match. This may hold even if the KB has reasonable string matches for the name itself. For example, the Nigerian saxophonist Peter Udo, who played with the Orchestra Baobab, is not included in any major KB (to the best of our knowledge), but there are many matches for the string “Peter Udo” as this is also a German first name. A good EL method needs to calibrate its linking decisions to avoid spurious choices, and should map mentions with low confidence in being KB entities to *null*. Such long-tail entities may become candidates to be included later in the KB life-cycle. We will revisit this issue in Chapter 8 on KB curation, specifically Section 8.6.3.

5.1.3 Coreference Resolution (CR)

The initial KB against which EL methods operate is inevitably incomplete, regarding both coverage of entities and coverage of different names for the known entities. The former is addressed by awareness of out-of-KB entities. The latter calls for grouping mentions into equivalence classes that denote the same entities. In NLP, this task is known as **coreference resolution (CR)**; in the world of structured data, its counterpart is the **entity matching (EM)** problem (see Section 5.2).

The CR task is highly related to the EL setting, as illustrated by Figure 5.2. Here, the input text contains underdetermined phrases like “the album”, “the singer” and “wife” (or say “his wife”). Longer texts will likely contain pronouns as well, such as “she”, “her”, “it”, “they”, etc. All these are not immediately linkable to KB entities. However, we can first aim to identify to which other mentions these coreferences refer, this way computing *equivalence classes of mentions*. In Figure 5.2, possible groupings are indicated by edges between mentions, and the correct ones are marked by thick lines in blue.

The ideal output would thus state that “Desire” and “the album” denote the same entity, and by linking one of the two mentions to the Bob Dylan album `Desire (album)`, EL covers both mentions. In general, however, the grouping and linking will be partial, meaning that some coreferences may be missed and some of the coreference groups may still be unlinkable – either because of remaining uncertainty or because the proper entity does not exist in the KB. Although this partial picture may look unsatisfying, it does give valuable information for KB construction and completion:

- Mentions in the same coreference group linked to a KB entity may be added as alias names, or simply textual cues, for an existing entity.
- Coreference groups that cannot be linked to a KB entity can be captured as candidates for new entities to be added, or at least reconsidered, later.

For example, if we pick up mentions like “Peter Udo”, “the sax player”, “the Nigerian saxophonist” and “he” as a coreference group, not only can we assert that he is not an existing KB entity, but we already have informative cues about what type of entity this is and even a gender cue.

Methods for coreference resolution over text inputs, and for coupling this with entity linking, can be rule-based (see, e.g., [471, 313, 143]), based on CRF-like graphical models (see, e.g., [142]) or based on neural learning (see, e.g., [100, 315, 272]). The latter benefits

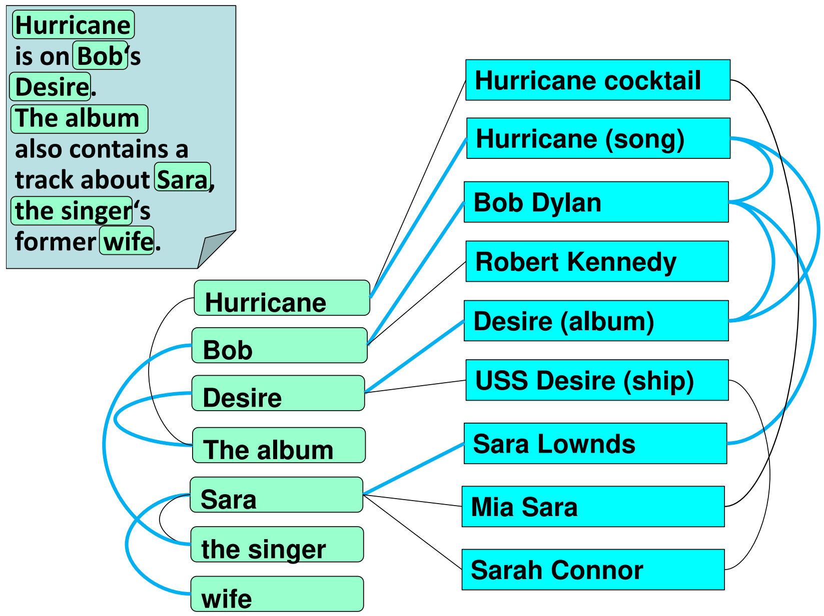


Figure 5.2: Example for Combined Entity Linking and Coreference Resolution

from feeding large unlabeled corpora into the model training (via embeddings such as BERT (e.g., [272], see also Section 4.5).

The CR task mostly focuses on short-distance mentions, often looking at successive sentences or paragraphs only. However, the task can be extended to compute coreference groups across distant paragraphs or even across different documents. This aims to mark up an entire corpus with equivalence classes of mentions, and partial linking of these classes to KB entities. The extended task is known as **cross-document coreference resolution (CCR)**, first studied by [25]. Methods along these lines include inference with CRF-like graphical models (e.g, [534]) and hierarchical clustering (e.g., [144]).

Benefit of CR for KB construction:
Partial linking of mentions to KB entities helps coreference resolution by capturing distant signals. Conversely, having good candidates for coreference groups is beneficial for EL, as it provides richer context for individual mentions. In addition to these dual benefits, coreference groups can be important for extracting types and properties of entities, when the latter are expressed with pronouns or underdetermined phrases (e.g., "the album"). For example, when given the text snippet

"Hurricane was not exactly a big hit.
It is a protest song about racism.",

we can infer the type `protest song` and its super-type `song` only with the help of CR. Likewise, in the example of Figure 5.2, we can acquire knowledge about Bob Dylan's ex-wife only when considering coreferences. So CR can improve extraction recall and thus the coverage of knowledge bases.

5.2 Entity Matching (EM)

Computing equivalence classes over a set of observed entity mentions is also a frequent task in data cleaning and integration. For example, when faced with two or more semantically overlapping databases or other datasets (incl. web tables), we need to infer which records or table rows correspond to the same entity. This task of **entity matching (EM)** or **duplicate detection**, historically called **record linkage**, is a long-standing problem in computer science [140, 160].

In a nutshell, all EM methods leverage cues from comparing records in a matching-candidate pair:

- **Name similarity:** The higher the string similarity between two names, the higher the likelihood that they denote matching entities (e.g., strings like "Sara" and "Sarah" being close).
- **Context similarity:** As context of a database record or table cell that denote an entity of interest, we should consider the full record or row as the data in such proximity often denote related entities or salient attribute values. Additionally, it can be beneficial to *contextualize* the mention in a table cell by considering the other cells in the same column, for example, as signals for the specific entity type of a cell (e.g., when all values in a table column are song titles).
- **Consistency constraints:** When a pair of rows from two tables is matched, this rules out matching one of the two rows to a third one, assuming that there are no duplicates within each table. With duplicates or when we consider more than two tables as input, matchings need to satisfy the transitivity of an equivalence relation.
- **Distant knowledge:** By connecting highly related entities, using a background KB, we can establish matching contexts despite low scores on string similarity. For example, a singer's name (e.g., Mick Jagger) and the name of his or her band (e.g., Rolling Stones) could be very different, but there is nevertheless a strong connection (e.g., in matching a Stones song, using either one or both of the related names).

These ingredients can be fed into matching rules generated from human-provided samples (e.g., [284, 533]), or used as input to supervised learning or probabilistic graphical models (e.g., [537, 40, 624, 475, 559, 291]) or neural networks (e.g., [405, 675]).

Similar techniques can be applied also to spotting matching entity pairs across datasets in the Web of (Linked) Open Data [225, 385]; see the survey by [419] on this *link discovery* problem. For query processing, the problem takes the form of finding joinable tables and the respective join columns and values (see, e.g., [671, 318]).

As EM may operate over large databases with millions of records, it faces a *scalability challenge*: conceptually comparing all record pairs from two databases may entail quadratic complexity and is bound to be intractable in practice. To overcome this potential show-stopper, the input needs to be partitioned into manageable blocks, based on a variety of **blocking techniques**. The simplest approach is to partition the records of both sides by the name string of the entities of interest (e.g., person names, company names, or songs or books, etc.) or by one or more of the most informative attributes (e.g., birthdate, address, etc.). More advanced techniques pre-compute fingerprints that approximate string similarity scores, such as min-hash sketches for n-gram overlap (and edit distance) [60, 82], and use these for partitioning. Because of these techniques, blocking-based EM algorithms are also referred to as *similarity joins* or *fuzzy joins*. The actual EM step then applies more refined techniques to compare records between blocks in the same partition. State-of-the-art methods do not rely on a single partitioning, but use adaptive and iterative blocking, so as to reduce the false negative rate and boost the overall accuracy (e.g., [623, 135, 96]).

Entity matching between databases or other structured data repositories is of interest for KB construction when incorporating premium sources into an existing KB. For example, when adding entities from GeoNames to a Wikipedia-derived KB, the detection of duplicates (to be eliminated from the ingest) is essentially an EM task. There is ample literature on EM methods for integrating heterogeneous databases and, recently, in the context of so-called **data lakes** [385]. Therefore, we do not discuss methods in more depth and instead refer to surveys and best-practice papers by [287, 416, 94, 135, 96].

5.3 Popularity, Similarity and Coherence Measures

All EL methods are based on quantitative measures for mention-entity popularity, mention-entity similarity and entity-entity coherence. This section elaborates on these measures.

Throughout the following, we assume that many entities in the KB, and especially the prominent ones, are uniquely identified also in Wikipedia. It is very easy for a KB to align its entities with Wikipedia articles. This holds for large KBs like Wikidata, even if the KB itself is not built from Wikipedia as a premium source. We will use Wikipedia articles as a background asset for various aspects of EL methods.

5.3.1 Mention-Entity Popularity

As most texts, like news, books and social media, are about prominent entities, the popularity of an entity determines a *prior likelihood* to be selected by an EL algorithm. For example, the Web contents about Elvis Presley is an order of magnitude larger than about the less known musician Elvis Costello. Therefore, when EL sees a mention "Elvis", the probability that this denotes `Elvis Presley` is a priori much higher than for Elvis Costello. To measure the **global popularity** of entities, a variety of indicators can be used:

- the length of an entity's Wikipedia article (or "homepage" in domain-specific platforms such as IMDB for movies or Goodreads for books,
- the number of incoming links of the Wikipedia page,
- the number of page visits based on Wikipedia usage statistics,
- the amount of user activity, such as clicks or likes, referring to the entity in a domain-specific platform or in social media, and more.

While considering global popularity is useful, it can be misleading and insufficient as a prior probability. For example, for the mention "Trump", the most likely entity is `Donald Trump`, but for the mention "Donald", albeit Donald Trump still being a candidate, the more likely entity is `Donald Duck`. This suggests that we should consider the *combination of mention and entity* for estimating popularity.

The most widely used estimator for mention-entity popularity is based on Wikipedia links [383, 386], exploiting the observation that href anchor texts are often short names and the pages to which they link are canonicalized entities already.

Link-based Mention-Entity Popularity:
The **mention-entity popularity score** for mention m and entity e is proportional to the occurrence frequency of hyperlinks with href anchor text *"m"* that point to the main page about e. This includes redirects within Wikipedia as well as interwiki links between different language editions.

Obviously, this works only for entities that have Wikipedia articles, and it hinges on sufficiently many href links pointing to these articles. On the other hand, the popularity score is useful only for prominent entities and will not be a good signal for long-tail entities anyway. Nevertheless, the approach can be generalized for larger coverage, by considering all kinds of Web pages with links to Wikipedia [550]. Further alternatives are to leverage query-click logs, by considering names in search-engine queries as mentions and subsequent clicks on Wikipedia articles or other kinds of homepages as linked entities.

5.3.2 Mention-Entity Context Similarity

The most obvious cue for inferring that mention m denotes entity e is to compare their *surface strings*. For m, this is given by the input text, for example, "Trump" or "President Trump". For e, we can consider the preferred (i.e., official or most widely used) label for the entity, such as "Donald Trump" or "Donald John Trump", but also alias names that are already included in the KB, such as "the US president". **String similarity between names**, like edit distance or n-gram overlap, can then score how good e is a match for m, this way ranking the candidate entities. In doing this, we can consider weights for tokens at the word or even character level. The weights can be derived from frequency statistics, in the spirit of IR-style *idf* weights. The weights and the similarity measure can be type-specific or domain-specific, dealing differently with say person names, organization acronyms, song titles, etc.

Beyond the basic comparison of m and names for e, a fairly obvious approach is to leverage the **mention context**, that is, the text surrounding m, and compare it to concise descriptions of entities or other kinds of **entity contextualization**. The mention context can take the form of a single sentence, single paragraph or entire document, possibly in weighted variants (e.g., weights decreasing with distance from m). The entity context depends on the richness of the existing KB. Entities can be augmented with descriptions taken, for example, from their Wikipedia articles (e.g., the first paragraph stating most salient points). Alternatively, the types of entities, their Wikipedia categories and other prominently appearing entities in a Wikipedia article (i.e., outgoing links) can be used for a contextualized representation. In essence, we create a pseudo-document for each candidate entity, and we may even expand these by external texts such as news articles about entities, or highly related concepts for the entity types using WordNet and other sources.

In the example of Figure 5.1, the mention “Hurricane” has words like “track” and “played” in its proximity, and the mention “Washington” is accompanied by the word “film”. These should be compared to entity contexts with words like song, music etc. for `Hurricane (song)` versus beverage, alcohol, rum etc. for `Hurricane (cocktail)`.

The following are some of the widely used measures for scoring the **mention-entity context similarity**.

Bag-of-Words Context Similarity:
Both m and e are represented as *tf-idf* vectors derived from bags of words (BoW). Their similarity is the scalar product or cosine between these vectors:

$$sim\,(cxt(m), cxt(e)) = \overrightarrow{BoW(cxt(m))} \cdot \overrightarrow{BoW(cxt(e))}$$

or analogously for cosine. Some variants restrict the BoW representation to *informative keywords* from both contexts, using statistical and entropy measures to identify the keywords.

A generalization of keyword-based contexts is to focus on **characteristic keyphrases** [238]: multi-word phrases that are salient and specific for candidate entities. Keyphrase candidates can be derived from entity types, category names, href anchor texts in an entity’s Wikipedia article, and other sources along these lines. For example, `Rubin Carter` (as a candidate for the mention “Carter” in Figure 5.1) would be associated with keyphrases such as “African-American boxer”, “people convicted of murder”, “overturned convictions”, "racism victim", "nickname The Hurricane" etc. Such phrases can be gathered from noun phrases n in the Wikipedia page (or other entity description) for e, and then scored and filtered by criteria like *pointwise mutual information (PMI)*:

$$weight(n|e) \sim \log \frac{P[n,e]}{P[n] \cdot P[e]}$$

or other information-theoretic entropy measures, with probabilities estimated from (co-) occurrence frequencies (in Wikipedia). Intuitively, the best keyphrases for entity e should frequently co-occur with e, but should not be globally frequent for all entities. The context of e, $cxt(e)$, then becomes the weighted set of keyphrases n for which $weight(n|e)$ is above some threshold.

Comparing a set of phrases against the mention context is a bit more difficult than for the Bag-of-Words representations. The reason is that exact matches of multi-word phrases are infrequent, and we need to pay attention to similar phrases with some words missing, different word order, and other variations. [238] proposed a word-proximity-aware window-based model for such approximate matches. For example, the keyphrase "racism victim" can be matched by "victim in a notorious case of racism".

Keyphrase Context Similarity:
Representing $cxt(e)$ as a weighted set KP of keyphrases n and $cxt(m)$ as a sequence of words conceptually broken down into a set W of (overlapping) small text windows of bounded length, the similarity is computed by aggregating the following scores:

- for each $n \in KP$ identify the best matching window $\omega \in W$;
- for each such ω compute the (sub-)set of words $w \in n \cap \omega$ and their maximum positional distance δ in ω;
- aggregate the word matches for n in ω with consideration of δ and the entity-specific weight for w (treating w as if it were a keyphrase by itself);
- aggregate these per-keyphrase scores over all $n \in KP$.

This template can be varied and extended in a number of ways.

With the advent of latent embeddings (see Subsection 4.5), both BoW and keyphrase models may seem to be superseded by word2vec-like and other kinds of embeddings, which implicitly capture also synonyms and other strongly related terms. However, at the word level alone, the embeddings are susceptible to over-generalization and drifting focus. For example, the word embedding for "Hurricane" brings out highly related terms that have nothing to do with the song or the boxer (e.g., "storm", "damage", "deaths" etc.). So it is important to use entity-centric embeddings likes the ones for wikipedia2vec [638], and ideally, these should reflect multi-word phrases as well.

Embedding-based Context Similarity:
With embedding vectors $\overrightarrow{cxt(m)}$ and $\overrightarrow{cxt(e)}$, the context similarity between m and e is $cos\left(\overrightarrow{cxt(m)}, \overrightarrow{cxt(e)}\right)$.

Each of these context-similarity models has its sweet spots as well as limitations. Choosing the right model thus depends on the topical domain (e.g., business vs. music) and the language style of the input texts (e.g., news vs. social media).

5.3.3 Entity-Entity Coherence

Whenever an input text contains several mentions, EL methods should compute their corresponding entities *jointly*, based on the principle that co-occurring mentions usually map to semantically coherent entities. To this end, we need to define measures for **entity-entity relatedness** that capture this notion of coherence.

One of the most powerful and surprisingly simple measures exploits the rich link structure of Wikipedia. The idea is to consider two entities as highly related if the hyperlink sets of their Wikipedia pages have a large overlap. More specifically, the incoming links are a good signal. For example, two songs like "Hurricane" and "Sara" are highly related because they are both linked to from the articles about `Bob Dylan`, `Desire (album)` and more (e.g., the other involved musicians). Likewise, `Bob Dylan` and `Elvis Presley` are notably related as there are quite a few Wikipedia categories that link to both of them. Intuitively, in-links are considered more informative than out-links as outgoing links tend to refer to more general entities and concepts whereas incoming links often have the direction from more general to more specific. These considerations have given rise to the following *link-based* definition of entity-entity coherence.

Link-based Entity-Entity Coherence:
For two entities e and f with Wikipedia articles that have incoming-link sets $In(e)$ and $In(f)$, their coherence score is

$$1 - \frac{\log\left(max\{|In(e)|, |In(f)|\}\right) - \log|In(e) \cup In(f)|)}{\log(|U|) - \log\left(min\{|In(e)|, |In(f)|\}\right)}$$

where U is the total set of known entities (e.g., Wikipedia articles about named entities).

This approach was pioneered by [386], and is thus sometimes referred to as the Milne-Witten metric. Similarly to link-based popularity measures, we can generalize this Wikipedia-centric link-overlap model to other settings. In a search engine's query-and-click log, entities whose pages both appear in the clicked-pages set for the same queries (so-called "co-clicks") should have a high relatedness score. In domain-specific content portals and web communities, such as Goodreads and LibraryThing for books or IMDB and Rottentomatoes for movies, the overlap of the user sets who expressed liking the same entity is a good measure for entity-entity relatedness. All these can be seen as instantiations of the **strong co-occurrence principle** (see Subsection 4.2).

When entities are represented as weighted sets of keywords (BoW) or weighted sets of keyphrases (KP), their relatedness can be captured by measures for the (weighted) overlap of these sets.

BoW-based and KP-based Entity-Entity Coherence:
Consider two entities e and f with associated keyword sets E and F whose entries x have weights $w_E(x)$ and $w_F(x)$, respectively. The coherence between e and f can be measured by the weighted Jaccard metric:

$$\sum_{x \in E \cap F} \frac{min\{w_E(x), w_F(x)\}}{max\{w_E(x), w_F(x)\}}$$

The extension to keyphrases is more sophisticated. It needs to consider also partial matches between keyphrases for e and for f (e.g., "rock and roll singer" vs. "rock 'n' roll musician"), following the same ideas as the KP-based context similarity model (see [238] for details).

Analogously to the context-similarity aspect, embeddings are a strong alternative for entity-entity coherence as well. They are straightforward to apply.

Embedding-based Entity-Entity Coherence:
With embedding vectors $\overrightarrow{e}$ and $\overrightarrow{f}$ for entities e and f, their relatedness for EL coherence is $cos(\overrightarrow{e}, \overrightarrow{f})$.

All these purely text-based coherence measures – keywords, keyphrases, embeddings – have the advantage that they can be computed solely from textual entity descriptions. There is no need for Wikipedia-style links, and not even for any relations between entities. This setting has been called *EL with a linkless KB* in [336]. That prior work, which predates embedding-based methods, made use of latent topic models (in the style of LDA [48]) for linkless EL. Today, word2vec-style embeddings or even BERT-like language models [118] (see Section 4.5) seem to be the more powerful choice, but they all fall into the same architecture presented above.

Yet another way of defining and computing entity-entity relatedness is by means of *random walks* over an existing knowledge graph (e.g., the link structure of Wikipedia), see, for example, [197]. Here each entity is represented by the (estimated) probability distribution of reaching related entities by random walks with restart. The relatedness score between entities can be defined as the relative entropy between two distributions.

The above are major cases within a wider space of measures for entity-entity coherence. A good discussion of the broader design space and empirical comparisons can be found in [557, 75, 455]. Ultimately, however, the best choice depends on the domain of interest (e.g., business vs. music) and the style of the ingested texts (e.g, news vs. social media).

5.4 Optimization and Learning-to-Rank Methods

All EL methods aim to optimize a scoring or ranking function that maximizes a combination of mention-entity popularity, mention-entity context similarity and entity-entity coherence. This can be formalized as follows.

EL Optimization Problem:
Consider an input text with entity mentions $M = \{m_1, m_2 \ldots\}$ each of which has entity candidates, $E(m_i) = \{e_{i1}, e_{i2} \ldots\}$, together forming a pool of target entities $E = \{e_1, e_2 \ldots\}$. The goal is to find a, possibly partial, function $\phi : M \rightarrow E$ that maximizes the objective

$$\alpha \sum_{m} pop(m, \phi(m)) + \beta \sum_{m} sim(cxt(m), cxt(\phi(m)))+$$

$$\gamma \sum_{e,f} \{coh(e, f) \mid \exists m, n \in M : m \neq n, e = \phi(m), f = \phi(n)\}$$

where α, β, γ are tunable hyper-parameters, *pop* denotes mention-entity popularity, *cxt* the context of mentions and entities, *sim* the contextual similarity and *coh* the pair-wise coherence between entities.

In principle, coherence could even be incorporated over the set of all entities in the image of ϕ, but the practical sweet spot is to break this down into pair-wise terms which allow more robust estimators.

Combinatorially, the function ϕ is the solution of a combined *selection-and-assignment* problem: selecting a subset of the entities and assigning the mentions onto them. We aim for the *globally best* solution, with a choice for ϕ that bundles the linking of all mentions together. Algorithmically, this optimization opens up a wide variety of design choices: from unsupervised scoring to graph-based algorithms all the way to neural learning. The basic choice, found in the classical works of [120, 67, 383, 103, 386, 379], is to view EL as a **local optimization** problem (notwithstanding its global nature): for each mention in the input text, we compute its best-matching entity from a pool of candidates. This is carried out for each mention independently of the other mentions, hence the adjective *local*. By this restriction, coherence is largely disregarded, but some aspects can still be incorporated via clever ways of contextualization. Most notably, the method of [386] first identifies *all unambiguous mentions* and expands their contexts by (the descriptions of) their respective entities. With this enriched context, the method then optimizes a combination of popularity and similarity (i.e., setting γ to zero in the general problem formulation). Generalized techniques for enhancing mention context and entity context via document retrieval have been devised by [336]. The literature on EL contains further techniques along these lines.

These approaches can be used to score and rank mention-entity pairs, but can likewise be broken down into their underlying scoring components as features for *learning* a ranker. Such methods have been explored using support vector machines and other kinds of classifiers, picking the entity with the highest classification confidence (e.g., [67, 386, 138, 525, 312]). Alternatively, more advanced **learning-to-rank (LTR)** regression models [350, 329], have been investigated (e.g., [669, 477, 197]), for example, with learning from pairwise preferences between entity candidates for the same mention. Note that these supervised learning methods hinge on the availability of a labeled training corpus where mentions are marked up with ground-truth entities.

5.5 Collective Methods based on Graphs

By factoring entity-entity relatedness scores into the context of candidate entities, context-similarity methods already go some way in considering global coherence, examples being [103, 104]. Nevertheless, more powerful EL methods make decisions on mapping mentions to entities *jointly* for all mentions of the input text. This family of methods is referred to as **Collective Entity Linking**. Many of these can be seen as operating on an **EL candidate graph**:

For an entity linking task, the **EL Candidate Graph** consists of

- a set M of mentions and a set E of entities as nodes,
- a set ME of *mention-entity edges* with weights derived from popularity and similarity scores, and
- a set EE of *entity-entity edges* with weights derived from relatedness scores between entities.

Figure 5.1, in Section 5.1, showed an example for such a candidate graph, with edge weights omitted.

5.5.1 Edge-based EL Methods

[161] proposed a relatively simple but effective and highly efficient approach for incorporating entity-entity edge weights into the scoring of mention-entity edges. Given a candidate mapping $m \mapsto e$, its score is augmented by the coherence of e with *all* candidate entities for

all other mentions in the graph:

$$score(m \mapsto e) = \cdots + \sum_{n \neq m} \sum_{f:(n,f) \in ME} sim(n,f) \cdot coh(e,f)$$

The intuition here is that a candidate e for m is rewarded if e has highly weighted edges or paths with all other entities in the entire candidate graph.

Obviously, the graph still contains spurious entities, but we expect those to have low coherence with others anyway. Conversely, entities for unambiguous mentions and entities with tight connections to many others in the graph have a strong influence on the decisions for all mentions. Hence the collective flavor of the method.

5.5.2 Subgraph-based EL Methods

Motivated by these considerations, a generalization is to consider entire *subgraphs* that connect the best cues for joint mappings. This leads to a powerful framework, although it entails more expensive algorithms. More specifically, we are interested in identifying *dense subgraphs* in the candidate graph such that there is at most one mention-entity edge for each mention (or exactly one if we insist on linking all mentions). Density refers to high edge weights, where both mention-entity weights and entity-entity weights are aggregated over the subgraph. We assume the weights of the two edge types are calibrated before being combined (e.g., via hyper-parameters like α, β, γ).

EL based on Dense Subgraph:
Given a candidate graph with nodes $M \cup E$ and weighted edges $ME \cup EE$, the goal is to compute the densest subgraph S, with nodes $S.M \subseteq M$ and $S.E \subset E$ and edges $S.ME \subset ME$ and $S.EE \subset EE$, maximizing

$$aggr\{weight(s) \mid s \in S.ME \cup S.EE\}$$

subject to the constraint:
for each $m \in M$ there is at most one $e \in S.E$ such that $(m,e) \in S.ME$. In this objective, *aggr* denotes an aggregation function over edge weights, a natural choice being summation (or the sum normalized by the number of nodes or edges in the subgraph).

Note that the resulting subgraph is not necessarily connected, as a text may be about different groups of entities, related within a group but unrelated across. However, this situation should be rare, and the method could enforce a connected subgraph. Also, it may be desirable to have identical mentions in a document mapped to the same entity (e.g., all mentions "Carter" linked to the same person), at least when the text is not too long. This can be enforced by an additional constraint.

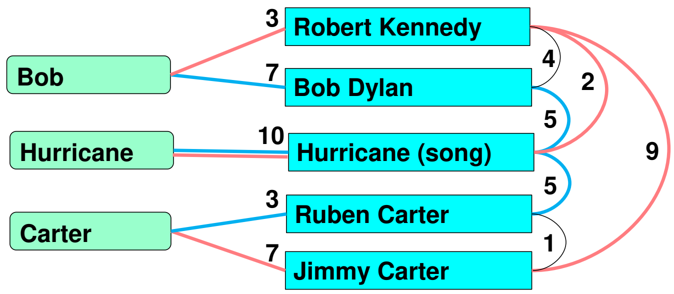


Figure 5.3: Example for Dense Subgraphs in EL Candidate Graph

Figure 5.3 illustrates this subgraph-based method with a simple example: 3 mentions and 5 candidate entities, with one mention being unambiguous. There are two subgraphs with high values for their total edge weight, shown in blue and red. The one in blue corresponds to the ground-truth mapping (assuming the context is still the Bob Dylan song), with a total weight of 30. The one in red is an alternative solution (centered on the two prominent politicians, one of which was a strong advocate against racism), with a total edge weight of 31. Both are substantially better than other choices, such as mapping the three mentions to Robert Kennedy, the song and Ruben Carter, which has a total edge weight of 23. However, the best subgraph by using sum for weight aggregation is the wrong output in red, albeit by a tiny margin only.

This observation motivates the following alternative choice for the aggregation function *aggr*: within the subgraph of interest, instead of paying attention only to the total weight, we consider the *weakest link*, that is, the edge with the lowest weight. The goal then is to maximize this minimum weight, making the weakest link as strong as possible. Using this objective, the best subgraph in Figure 5.3 is the one with Bob Dylan, the song and Ruben Carter, as the lowest edge weight in the respective subgraph is 3, whereas it is 2 for the alternative subgraph with Robert Kennedy, the song and Jimmy Carter.

EL based on Max-Min Subgraph:
For a given candidate graph, the goal is to compute the subgraph S that maximizes

$$min\{weight(s) \mid s \in S.ME \cup S.EE\}$$

subject to the constraint:
for each $m \in M$ there is at most one $e \in S.E$ such that $(m, e) \in S.ME$.

Both variants of the dense-subgraph approach are NP-hard, due to the constraint about mention-entity edges. However, there are good approximation algorithms, including greedy removal of weak edges as well as stochastic search to overcome local optima. This family of methods has been proposed by [241], and achieved good results on benchmarks with news articles, in a completely unsupervised way (except for tuning a small number of hyper-parameters). Methods with similar considerations on incorporating coherence have been developed by [477, 525].

5.5.3 EL Methods based on Random Walks

Using the same EL candidate graph as before, an alternative approach that also has efficient implementations, is based on **random walks with restarts**, essentially the same principle that underlies *Personalized Page Rank* [219].

First, edge weights in the candidate graph are re-calibrated to become proper transition probabilities, and a small restart probability is chosen to jump back to the starting node of a walk. Conceptually, we initiate such walks on each of the mentions, making probabilistic decisions for traversing both mention-entity and entity-entity edges many times, and occasionally jumping back to the origin. In the limit, as the walk length approaches infinity, the visiting frequencies of the various nodes converge to *stationary visiting probabilities*, which are then interpreted as scores for mapping mentions to entities. An actual implementation would bound the length of each walk, but walk repeatedly to obtain samples towards better approximation. Alternatively, iterative numeric algorithms from linear algebra, most notably, Jacobi iteration, can be applied to the transition matrix of the graph, until some convergence criterion is reached for the best entity candidates of every mention.

EL based on Random Walks with Restart:
Given a candidate graph with weights, re-calibrate the weights into proper transition probabilities.
For each mention $m \in M$

- approximate the visiting probabilities of the possible target entities $e \in E$: $(m, e) \in ME$, and
- map m to the entity e with the highest probability.

Although these algorithms make linking decisions one mention at a time, they do capture the essence of collective EL as the walks involve the entire candidate graph and the stationary visiting probabilities take the mutual coherence into account. EL methods based on random walks and related techniques include, for example, [399, 196, 451, 190].

5.5.4 EL Methods based on Probabilistic Graphical Models

The coherence-aware graph-based methods can also be cast into **probabilistic graphical models**, like CRFs and related models. They can be seen as reasoning over a joint probability distribution

$$P[m_1, m_2 \ldots, e_1, e_2 \ldots, d]$$

with mentions m_i, entities e_j and the context given by document d. This denotes the likelihood that a document d contains entities $e_1, e_2 \ldots$ and that these entities are textually expressed in the form of mentions $m_1, m_2 \ldots$. Obviously, this high-dimensional distribution is not tractable. So it is factorized by making model assumptions and mathematical transformations, such as

$$P[m_1, m_2 \ldots, e_1, e_2 \ldots, d] = \prod_{i,j} P[m_i | e_j, d] \cdot \prod_{j,k} P[e_j, e_k]$$

where $P[m_i|e_j, d]$ is the probability of m_i expressing e_j in the context of d and $P[e_j, e_k]$ is the probability of the two entities co-occurring in the same, semantically meaningful document.

This kind of probabilistic reasoning can be cast into a **CRF model** or **factor graph** (cf. Section 4.4) as follows.

CRF Model for Entity Linking:
For each mention m_i with entity candidates $E_i = \{e_{i1}, e_{i2} \ldots\}$, the model has a **random variable** X_i with values from E_i. These variables capture the probabilities $P[m_i|e_j]$.
For each candidate entity e_k (for any of the mentions), the model has a binary **random variable** Y_k that is true if e_k is mentioned in the document and false otherwise. These variables capture probabilities $P[e_k|d]$ of entity occurrence in the document.
All variables are assumed to be conditionally independent, except for the following **coupling factors**:

- X_i, Y_j are coupled if e_j is a candidate for m_i,
- Y_j, Y_k are coupled for all pairs e_j, e_k.

Figure 5.4 depicts an example, showing how the candidate graph for our running example can be cast into the CRF structure with variables for each mention and entity node, and coupling factors for each of the edges.

Unlike the CRF models for NER, discussed in Section 4.4, this kind of CRF does not operate on token sequences but on graph-structured input. So it falls under a more general class of **Markov Random Fields (MRF)**, but the approach is nevertheless widely referred to as a CRF model. The joint distribution of all X_i and Y_j variables is factorized, by the Markov assumption, according to the cliques in the graph. This way, we obtain one "clique potential" or "coupling factor" per clique, often with restriction to binary cliques, coupling two variables (i.e., the edges in the graph). Training such a model entails learning per-clique weights (or estimating conditional probabilities for the coupled variables), typically using gradient-descent techniques. Alternatively, similarity and coherence scores can be used

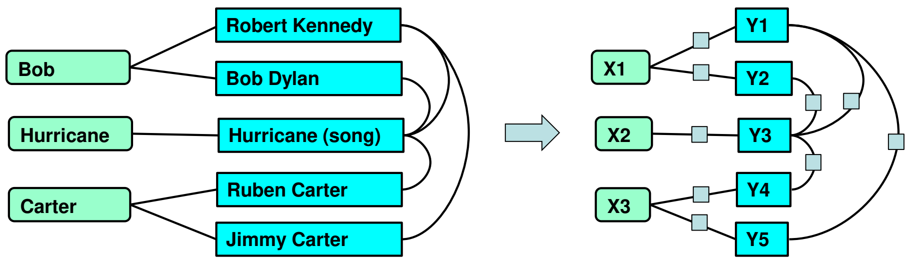


Figure 5.4: Example for CRF derived from EL candidate graph

for these purposes, at least for a good initialization of the gradient-descent optimization. Inference on the values of variables, given a new text document as input, is usually limited to joint MAP inference: computing the combination of variable values that has the highest posterior likelihood. This involves Monte Carlo sampling, belief propagation, or variational calculus (cf. Section 4.4 on CRFs for NER).

CRF-based EL has first been developed by [297], with a variety of enhancements in follow-up works such as [142, 178, 422].

Many CRF-like models can also be cast into **Integer Linear Programs (ILP)**: a discrete optimization problem with constraints [500, 498].

ILP Model for Entity Linking:
For each mention m_i with entity candidates $E_i = \{e_{i1}, e_{i2} \dots\}$, the model has a binary **decision variable** X_{ij} set to 1 if m_i truly denotes e_j.
For each pair of candidate entities e_k, e_l (for any of the mentions), the model has a binary **decision variable** Y_{kl} set to 1 if e_k and e_l are indeed both mentioned in the input text.
The objective function for the ILP is to maximize the data evidence for the choice of 0-1 values for the X_{ij} and Y_{kl} variables, subject to constraints:

$$\text{maximize} \;\; \beta \sum_{ij} weight(m_i, e_j) X_{ij} \; + \; \gamma \sum_{kl} weight(e_k, e_l) Y_{kl}$$

with weights corresponding to similarity and coherence scores and hyper-parameters β, γ.
The constraints specify that mappings are functions, couple the X_{ij} and Y_{ij} variables, and may optionally capture transitivity among identical mentions or (obvious) coreferences, if desired:

- $\sum_j X_{ij} \leq 1$ for all i;
- $Y_{kl} \geq X_{ik} + X_{jl} - 1$, stating that Y_{kl} must be 1 if both e_k and e_l are chosen as mapping targets;
- $X_{ik} \leq X_{jk}$ and $X_{ik} \geq X_{jk}$ for all k for identical mentions m_i, m_j;
- $0 \leq X_{ij} \leq 1$ and $0 \leq Y_{kl} \leq 1$ for all i, j, k, l.

Solving such an ILP is computationally expensive: NP-hard in the worst case and also costly in practice for large instances. However, there are very efficient ILP solvers, such as Gurobi (https://www.gurobi.com/), which can handle reasonably sized inputs such as short news articles with tens of mentions and hundreds of candidate entities. Larger inputs could have their candidate space pruned first by other, simpler, techniques. Moreover, ILPs can be relaxed into LPs, linear programs with continuous variables, followed by randomized rounding. Often, this yields very good approximations for the discrete optimization (see also [297]).

5.6 Methods based on Supervised Learning

Early work on EL (e.g., [67, 386, 138, 477, 525, 104, 312]) already pursued machine learning for ranking entity candidates, building on labeled training data in the form of ground-truth mention-entity pairs in corpora (most notably, Wikipedia articles or annotated news articles). These methods used support vector machines, logistic regression and other learners, all relying on feature engineering, with features about mention contexts and a suite of cues for entity-entity relatedness. More recently, with the advent of deep neural networks, these feature-based learners have been superseded by end-to-end architectures without feature modeling. However, these methods still, and perhaps even more strongly, hinge on sufficiently large collections of training samples in the form of correct mention-entity pairs.

5.6.1 Neural EL

Recall from Section 4.4 that neural networks require real-valued vectors as input. Thus, a key point in applying neural learning to the EL problem is the *embeddings* of the inputs: mention context (both short-distance and long-distance), entity description and most strongly related entities, and more. This *neural encoding* is already a learning task by itself, successfully addressed, for example, by [169, 639, 640, 200]. The jointly learned embeddings are fed into a deep neural network, with a variety of architectures like LSTM, CNN, Feed-Forward, Attention learning, Transformer-style, etc. The output of the neural classifier is a scoring of entity candidates for each mention. For end-to-end training, a typical choice for the loss function is softmax over the cross-entropy between predictions and ground-truth distribution. Figure 5.5 illustrates such a neural EL architecture. As embedding vectors are fairly restricted in size, mention contexts can be captured at different scopes: short-distance like sentences as well as long-distance like entire documents. By the nature of neural networks, the "cross-talk" between mentions and entities and among entities is automatically considered, capturing similarity as well as coherence.

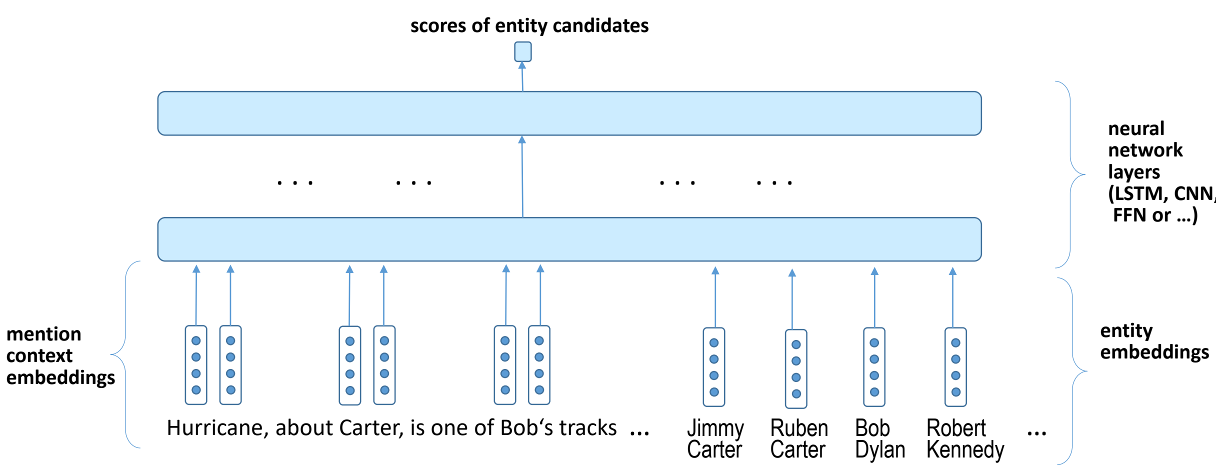


Figure 5.5: Illustration of Neural EL Architecture

Neural networks for EL are trained end-to-end (e.g., [151, 282, 406, 518, 252]) on labeled corpora where mentions are marked up with their proper entities, using gradient descent techniques. Some of these methods integrate EL with the NER task, jointly spotting mentions and linking them. In the literature, Wikipedia full-text with hyperlink targets as ground truth provides ample training samples. However, the articles all follow the same encyclopedic style. Therefore, the learned models, albeit achieving excellent performance on withheld Wikipedia articles, do not easily carry over to text with different stylistic characteristics and neither to domain-specific settings such as biomedical articles or health discussion forums. Another large resource for training is the *WikiLinks* corpus [535] which comprises Web pages with links to Wikipedia articles. This captures a wider diversity of text styles, but the ground-truth labels have been compiled automatically, hence containing errors.

Overall, it seems that neural EL is not yet as mature and successful as its neural counterparts for NER (see Section 4.4), as it is easier to obtain training data for NER. Neural EL shines when trained with large labeled collections and the downstream texts to which the learned linker is applied have the same general characteristics. When training samples are scarce or the use-case data characteristics substantially deviate from the training data, it is much harder for neural EL to compete with feature-based unsupervised methods. Very recently, methods for *transfer learning* have been integrated into neural EL (e.g., [357, 629]). These methods are trained on one labeled collection, but applied to a different collection which does not have any labels and has a disjoint set of target entities. A major asset to this end is the integration of large-scale language embeddings, like BERT [118], which covers both training and target domains. The effect is an implicit capability of "reading comprehension", which latently captures relevant signals about context similarity and coherence. Transfer learning still seems a somewhat brittle approaches, but such methods will be further advanced, leveraging even larger language embeddings, such as GPT-3 based on a neural network with over 100 billion parameters [62].

Regardless of future advances along these lines, we need to realize that EL comes in many different flavors: for different domains, text styles and objectives (e.g., precision vs. recall). Therefore, flexibly configurable, unsupervised methods with explicit feature engineering will continue to play a strong role. This includes methods that require tuning a handful of hyper-parameters, which can be done by domain experts or using a small set of labeled samples.

5.7 Methods for Semi-Structured Data

In addition to text documents, KB construction also benefits from tapping semi-structured contents with lists and tables. In the following, we focus on the case of ad-hoc web tables as input, to exemplify EL over semi-structured data. Table 5.1 shows an example with ambiguous mentions such as "Elvis", "Adele", "Columbia", "RCA" as well as abbreviated or slightly misspelled names (e.g., "Pat Garrett" should be the album `Pat Garrett & Billy`

Name	Title	Album	Label	Year
Bob Dylan	Hurricane	Desire	Columbia	1976
Bob Dylan	Sara	Desire	Columbia	1976
Bob Dylan	Knockin on Heavens Door	Pat Garrett	Columbia	1973
Elvis	Cant Help Falling in Love	Blue Hawaii	RCA	1961
Adele	Make You Feel My Love	n/a	XL	2008

Table 5.1: Example for entity linking task over web tables

`the Kid`). Note that such tables are usually surrounded by text – within web pages, with table captions, headings etc., which can be harnessed as additional context.

From a traditional database perspective, it seems that the best cues for EL over tables is to exploit the table schema, that is, column headers and perhaps inferrable column types. However, these tables are very different from well-designed databases: they are hand-crafted in an ad-hoc manner, and their column names are often not exactly informative (e.g., "Name", "Title", "Label" are very generic). Thus, it seems that the EL problem is much harder for tables. However, we can leverage the tabular structure to guide the search for the proper entities. Specifically, we pay attention to **same-row mentions** and **same-column mentions**:

- **Same-row mentions** are most tightly related. So their coherence should be boosted in the objective function.
- **Same-column mentions** are not directly related, but they are typically of the same type, such as `musicians`, `songs`, `music albums` and `record labels`. So the objective function should incorporate a soft constraint for per-column *homogeneity*.

By taking these design considerations into account, the EL optimization can be varied as follows.

EL Optimization for Web Tables:
Consider a table with c columns, r rows and entity mentions m_{ij} where i, j are the row and column where the mention occurs. Each m_{ij} has a set of entity candidates $E(m_{ij})$. All mentions together are denoted as M, and the pool of target entities overall as E. For ease of notation, we assume that all table cells are entity mentions (i.e., disregarding the fact that some columns are about literal values). The goal is to find a, possibly partial, function $\phi : M \to E$ that maximizes the objective

$$\alpha \sum_{m_{ij} \in M} pop(m_{ij}, \phi(m_{ij})) +$$

$$\beta 1 \sum_{m_{ij} \in M} sim(rowcxt(m_{ij}), cxt(\phi(m_{ij}))) +$$

$$\beta 2 \sum_{m_{ij} \in M} sim(doccxt(m_{ij}), cxt(\phi(m_{ij}))) +$$

$$\gamma \sum_{e,f \in E} \{coh(e, f) \mid m_{ij}, m_{ik} \in M : j \neq k, e = \phi(m_{ij}), f = \phi(m_{ik})\} +$$

$$\delta \sum_{j=1..c} hom\{type(e) \mid e = \phi(m_{*j}, * = 1..r\}$$

where $\alpha, \beta 1, \beta 2, \gamma$, δ are tunable hyper-parameters. As before, *pop* denotes mention-entity popularity, *cxt* the context of mentions and entities in two variants for mentions: *rowcxt* for same-row cells, *doccxt* for the entire document. *sim* denotes contextual similarity and *coh* pair-wise coherence between same-row entities. *hom* is a measure of type *homogeneity* and *specificity*.

This framework leaves many choices for the underlying measures: defining specifics of the two context models, defining the measures for type homogeneity and specificity, and so on. For example, *hom* may combine the fraction of per-column entities that have a common type and the depth of the type in the KB taxonomy. The latter is important to avoid overly generic types like `entity`, `person` or `artefact`. The inference of column types has been addressed as a problem by itself (e.g., [593, 85]). The case of *lists*, which can be viewed as single-column tables, has received special attention, too (e.g., [524, 228]).

The literature on EL over tables, most notably [338, 39, 253, 493], discusses a variety of viable design choices in depth. [660] is a recent survey on knowledge extraction from web tables.

Algorithmically, many of the previously presented methods for text-based EL carry over to the case of tables. Scoring and ranking methods can simply extend their objective functions to the above table-specific model. Graph-based methods, including dense subgraphs, random walks and CRF-based inference, merely have to re-define their input graphs accordingly.

The seminal work of [338] did this with a probabilistic graphical model, integrating the column type inference in a joint learning task.

Figure 5.6 illustrates this graph-construction step: edges denote CRF-like coupling factors or guide random walks (only some edges are shown). In the figure, the type homogeneity is depicted by couplings with the prevalent column type. Alternatively, the CRF could couple all mention pairs in the same column including the column header [39].

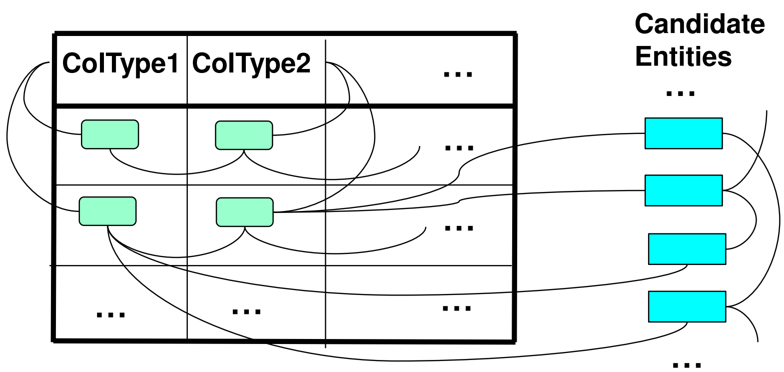


Figure 5.6: Illustration of EL Graph for Tables

5.8 Variations and Extensions

Iterative Linking:
A simple yet powerful principle that can be combined with virtually all EL methods is to make linking decisions in multiple rounds, based on the mapping confidence (see, e.g., [196, 421, 451]). Initially, only unambiguous mentions are mapped, unless there is uncertainty on whether they could denote out-of-KB entities. In the next round, only those mentions are mapped for which the method has high confidence. After every round, all similarity and coherence measures are updated triggering updates to the graph or other model on which EL operates. As more and more entities are mapped, they create a more focused context for subsequent rounds. For the running example, suppose that we can map “Hurricane” to the song with high confidence. Once this context about music is established, the confidence in linking “Bob” to Bob Dylan, rather than any other prominent Bobs, is boosted.

Domain-specific Methods:
There are numerous variations and extensions of EL methods, including *domain-specific* approaches, for example, for mentions of proteins, diseases etc. in biomedical texts (see, e.g., [17, 170, 108, 267] and references given there), and *multi-lingual* approaches where training data is available only in some languages and transferred to the processing of other

languages (see, e.g., [531] and references there). The case for domain-specific methods can also be made for music (e.g., names of songs, albums, bands – which include many common words and appear incomplete or misspelled), bibliography with focus on author names and publication titles, and even business where company acronyms are common and product names have many variants.

Methods for Specific Text Styles:
There are approaches customized to specific kinds of *text styles*, most notably, social media posts such as tweets (e.g., [351, 526, 116]) with characteristics very different from encyclopedic pages or news articles. Yet another specific kind of input is *search-engine queries* when users refer to entities by telegraphic phrases (e.g., "Dylan songs covered by Grammy and Oscar winners"). For EL over queries, [513] developed powerful methods based on probabilistic graphical models.

Methods for Specific Entity Types:
Finally, there are also specialized EL methods for disambiguating geo-spatial entities, such as "Twin Towers" (e.g., in Kuala Lumpur, or formerly in New York) and temporal entities such as "Mooncake Festival". Methods for these types of entities are covered, for example, by [321, 508, 83] for spatial mentions, aka. *toponyms*, and [555, 301] for temporal expressions, aka. *temponyms*. A notable project where spatio-temporal entities have been annotated at scale is the GDELT news archive (https://www.gdeltproject.org/), supporting event-oriented knowledge discovery [316].

5.9 Take-Home Lessons

Key points to remember from this chapter are the following:

- *Entity Linking (EL)* (aka. Named Entity Disambiguation) is the task of mapping entity mentions in web pages (detected by NER, see Chapter 4) onto uniquely identified entities in a KB or similar repository. This is a key step for constructing *canonicalized* KBs. Full-fledged EL methods also consider the case of *out-of-KB* entities, where a mention should be mapped to null rather than any entity in the KB.
- The input sources for EL can be *text documents* or *semi-structured contents* such as lists or tables in web pages. In the text case, related tasks like coreference resolution (CR) for pronouns and common noun phrases, or even general word sense disambiguation, may be incorporated as well.
- *Entity Matching (EM)* is a variation of the EL task where the inputs are structured data sources, such as database tables. The goal here is to map mentions in data records from one source to those of a different source and, this way, compute equivalence classes. There is not necessarily a KB as a reference repository.

- EL and EM leverage a variety of signals, most notably: a-priori *popularity* measures for entities and name-entity pairs, *context similarity* between mentions and entities, and *coherence* between entities that are considered as targets for different mentions in the same input. Specific instantiations may exploit existing links like those in Wikipedia, text cues like keywords and keyphrases, or embeddings for latent encoding of such contexts.
- EL methods can be chosen from a spectrum of paradigms, spanning *graph algorithms*, *probabilistic graphical models*, feature-based *classifiers*, all the way to feature-less *neural learning*. A good choice depends on the prioritization of complexity, efficiency, precision, recall, robustness and other quality dimensions.
- None of the state-of-the-art EL methods seems to universally dominate the others. There is (still) no one-size-fits-all solution. Instead, a good design choice depends on the application setting: scope and scale of entities under consideration (e.g., focus on vertical domains or entities of specific types), language style and structure of inputs (e.g., news articles vs. social media vs. scientific papers), and requirements of the use case (e.g., speed vs. precision vs. recall).

6 KB Construction: Attributes and Relationships

6.1 Problem and Design Space

Using methods from the previous chapters, we can now assume that we have a knowledge base that has a clean and expressive taxonomy of semantic types (aka. classes) and that these types are populated with a comprehensive set of canonicalized (i.e., uniquely identified) entities.

The next step is to enrich the entities with *properties* in the form of SPO triples, covering both

- *attributes* with literal values such as the birthday of a person, the year when a song or album was released, the maximum speed and energy consumption of a car model, etc., and
- *relations* with other entities such as birthplace, spouse, composers and musicians for a song or album, manufacturer of a car, etc.

In this chapter, we present methods for extracting such SPO triples; most of these can handle both attributes and relations in a more or less unified way. We will see that many of the principles (e.g., statement-pattern duality), key ideas (e.g., pattern learning) and methodologies (e.g., CRFs or neural networks) of the previous chapters are applicable here as well.

Assumptions:
Best-practice methods build on a number of assumptions that are justified by already having a clean and large KB of entities and types.

- **Argument Spotting:** Given input content in the form of a text document, Web page, list or table, we can spot and canonicalize arguments for the subject and object of a candidate triple. This assumption is valid because we already have methods for entity discovery and linking. As for attribute values, specific techniques for dates, monetary numbers and other quantities (with units) can be harnessed for spotting and normalization (e.g., [362, 502, 9, 555]).
- **Target Properties:** We assume that, for each type of entities, we have a fairly good understanding and a reasonable initial list of which properties are relevant to capture in the KB. For example, we should know upfront that for people in general we are interested in birthdate, birthplace, spouse(s), children, organizations worked for, awards, etc., and for musicians, we additionally need to harvest songs composed or performed, albums released, concerts given, instruments played, etc. These lists are unlikely to be complete, but they provide the starting point for this chapter. We will revisit and relax the assumption in Chapter 7 on the construction and evolution of open schemas.
- **Type Signatures:** We assume that each property of interest has a type signature

such that we know the domain and range of the property upfront. This is part of the KB schema (or ontology). By associating properties with types, we already have the domain, but we require also that the range is specified in terms of data types for both attributes (literal values) and relations (entity types). This enables high-precision knowledge acquistion, as we can leverage type constraints for de-noising. For example, we will assume the following specifications:

```
birthdate: person x date
birthplace: person x location
spouse: person x person
worksFor: person x organization
```

Schema Repositories of Properties:
Where do these pre-specified properties of interest and their type signatures come from? Spontaneously, one may think this is a leap of faith, but on second thought, there are major assets already available.

- Early KB projects like Yago and Freebase demonstrated that it is well feasible, with limited effort, to manually compile schemas (aka. ontologies) for relevant properties. *Freebase* comprised several thousands of properties with type signatures.
- Frameworks like *schema.org* [193] have specified *vocabularies* for types and properties. These are not populated with entities, but one can easily use the schemas to drive the KB population. Currently, schema.org comprises mostly business-related types (ca. 1000) and their salient properties.
- There are rich *catalogs* and *thesauri* that cover a fair amount of vocabulary for types and properties. Some are well organized and clean, for example, the icecat.biz catalog of consumer products. Others are not quite so clean, but can still be starting points towards a schema, an example being the UMLS thesaurus for the biomedical domain (https://www.nlm.nih.gov/research/umls/).
- *Domain-specific KBs*, say on food, health or energy, can start with some of the above repositories and would then require expert efforts to extend and refine their schemas. This is manual work, but it is not a huge endeavor, as the KB is very unlikely to require more than a few thousand types and properties. For health KBs, for example, some tens of types and properties already cover a useful slice (e.g., [150, 605]).

6.2 Pattern-based and Rule-based Extraction

6.2.1 Specified Patterns and Rules

The easiest and most effective way of harvesting attribute values and relational arguments, for given entities and a target property, is again to tap premium sources like Wikipedia (or IMDB, Goodreads etc. for specific domains). They feature entity-specific pages and their structure follows fairly rigid templates. Therefore, extraction patterns can be specified with relatively little effort, most notably, in the form of **regular expressions (regex)** over the text and existing markup of the target pages. The underlying assumption for the viability of this approach is:

> **Consistent Patterns in Single Web Site:**
> In a single web site, all (or most) pages about entities of the same type (e.g., musicians or writers) exhibit the same patterns to express certain properties (e.g., their albums or their books, respectively). A limited amount of diversity and exceptions needs to be accepted, though.

Figure 6.1: Examples of Wikipedia Infoboxes

6.2.1.1 Regex Patterns

Within Wikipedia, semi-structured elements like infoboxes, categories, lists, headings, etc. provide the best opportunity for harvesting facts by regular expressions. Consider the infoboxes shown in Figure 6.1 for three musicians (introducing new ones for a change, to give us a break from Bob Dylan and Elvis Presley). Our goal is to extract, say, the dates and places of birth of these people, to populate the `birthdate` attribute and `birthplace` relation. For these examples, the *Born* fields provide this knowledge, with small variations, though, such as showing only the year for Nive Nielsen or repeating the person name for Jimmy Page. The following regular expressions specify the proper extractions, coping with the variations. For simplicity of explanation, we restrict ourselves to the birth year and birth city.

```
birth year X: Born .* (X = (1|2)[0-9]{3}) .*
birth city Y: Born .* ([0-9]{4}|")") (Y = [A-Z]([a-z])+) .*
```

In these expressions, ".*" denotes a wildcard for any token sequence, "|" and "[...]" denote disjunctions and ranges of tokens, "{...}" and "+" are repetition factors, and putting ")" itself in quotes is necessary to distinguish this token from the parenthesis symbol used to group sub-structures in a regex. Note that the specific syntax for regex varies among different pattern-matching tools.

Intuitively, the regex for birth year finds a subsequence X that has exactly four digits and starts with 1 or 2 (disregarding, for simplicity, people who were born more than 1020 years ago). The regex for birth cities identifies the first alphabetic string that starts with an upper-case letter and follows a digit or closing parenthesis.

This is still not perfectly covering all possible cases. For example, cities could be multi-word noun phrases (e.g., New Orleans). We do not show more complex expressions for ease of explanation. It is straightforward to extend the approach for both i) completeness, like extracting the full date rather than merely the year and the exact place, and ii) diversity of showing this in infoboxes. On the latter aspect, the moderation of Wikipedia has gone a long way towards standardizing infobox conventions by templates, but it could still be (and earlier was the case) that some people's infoboxes show fields *birth place*, *place of birth*, *born in*, *birth city*, *country of birth*, etc. Nevertheless, it is limited effort to manually specify wide-coverage and robust regex patterns for hundreds of attributes and relations. The YAGO project, for example, did this for about 100 properties in a few days of single-person work [563]. Industrial knowledge bases harvest deterministic patterns from Web sites that are fed by back-end databases, such that each entity page has the very same structure (e.g., IMDB pages for the cast of movies).

6.2.1.2 Inducing Regex Patterns from Examples

To ease the specification of regex patterns, methods have been developed that merely require marking up examples of the desired output in a small set of pages, sometimes even supported by visual tools (e.g., [507, 162, 209]). For restricted kinds of patterns, it is then possible to automatically learn the regex or, equivalently, the underlying finite-state automaton, essentially inferring a regular grammar from examples of the language. This methodology applied to pattern extraction has become known as **wrapper induction** [300, 545, 299, 410, 36]. The survey [511] covers best-practice methods, with emphasis on CRF-based learning. Wrapper induction is a standard building block for information extraction today.

6.2.1.3 Type Checking

Neither manually specified nor learned regex patterns are perfect: there could always be an unanticipated variation among the pages that are processed. An extremely useful technique to prune out false positives among the extracted results is **semantic type checking**, utilizing the a-priori knowledge of type signatures for the properties of interest (see Section 6.1). If we expect `birthplace` to have cities as its range rather than countries or even continents, we can test an extracted argument for this relation against the specification. The types themselves can be looked up in the existing KB of entities and classes, after running entity linking on the extracted argument. This technique substantially improves the precision of regex-based KB population [563, 240]. It equally applies to literal values of attributes if there are pre-specified patterns, for example, for dates, monetary values or quantities with units.

6.2.1.4 Operator-based Extraction Plans

Often, a number of patterns, rules, type-checking and other steps have to be combined into an entire execution plan to accomplish some extraction task. The underlying steps can be seen as operators in an algebraic language. The **System T** project [485, 90, 89] has developed a declarative language, called AQL (Annotation Query Language), and a framework for orchestrating, optimizing and executing algebraic plans combining such operators. In addition to expressive kinds of pattern matching, the framework includes operators for text spans and for combining intermediate results. A similar project, with a declarative language called **Xlog**, was pursued by [522, 521], and closely related research was carried out by [50] and [260].

To illustrate the notion and value of operator-based execution plans, assume that we want to extract from a large corpus of web pages statements about music bands and their live concerts, specifically, the relation between the involved musicians and the instruments that they played. An example input could look as follows:

> **Led Zeppelin returns with rocking London reunion.**
> *The quartet had a crowd of around 20,000 at London's 02 Arena calling for more at the end of 16 tracks ranging from their most famous numbers to less familiar fare. Lead singer Robert Plant, 59, strutted his way through "Good Times Bad Times" to kick off one of the most eagerly-anticipated concerts in recent years. A grey-haired Jimmy Page, 63, reminded the world why he is considered one of the lead guitar greats, while John Paul Jones, 61, showed his versatility jumping from bass to keyboards. Completing the quartet was Jason Bonham on drums.*

We aim to extract all triples of the form

$$playsInstrument : musician \times instrument,$$

namely, the five SO pairs

`(Robert Plant, vocals)`, `(John Paul Jones, bass)`,
`(John Paul Jones, keyboards)`, `(Jimmy Page, guitar)`, `(Jason Bonham, drums)`.

In addition, we want to check that this really refers to a live performance. The extraction task entails several steps:

1. Detect all person names in the text.
2. Check that the mentions of people are indeed musicians, by entity linking and type checking, or accept them as out-of-KB entities.
3. Detect all mentions of musical instruments, using the instances from the KB type `music instruments`, including specializations such as `Gibson Les Paul` for `electric guitar` and paraphrases from the KB dictionary of labels, such as "singer" for `vocals`.
4. Check that the instrument mentions refer to specific mentions of musicians, for example, by considering only pairs of musician and instrument that appear in the same sentence. Here the text proximity condition may have to be varied depending on the nature and style of the text, for example, by testing for co-occurrences within a text span of 30 words, or by first applying co-reference resolution.
   Sometimes, even deeper analyis is called for, to handle difficult cases where subject and object co-occur incidentally without standing in the proper relation to each other.
5. Check that the entire text refers to a live performance. For example, this could require checking for the occurrence of a date and specific kinds of location like concert halls, theaters, performance arenas, music clubs and bars, or festivals.

There are many ways for ordering the execution of these steps, each of which involves sub-steps (e.g., for matching performance locations), or for running them in parallel or in a pipelined manner (with intermediate results streamed between steps). When applying such an entire operator ensemble to a large collection of input pages, the choice of order or parallel execution is crucial. The reason is that different choices incur largely different costs because they materialize intermediate results of highly varying sizes. The System T approach therefore views the entire ensemble as a declarative task, and invokes a query optimizer to pick the best execution plan. This involves estimating the selectivities of operators, that is, the fraction of pages that yield intermediate results and the number of intermediate candidates per page. For query optimization over text, this cost-estimation aspect is still underexplored, notwithstanding results by the SystemT project [485] as well as [522] and [258].

6.2.1.5 Patterns for Text, Lists and Trees

Extraction patterns can also be specified for texts or lists as inputs. This is often amazingly easy and can yield high-quality outputs. For example, many Wikipedia articles and other kinds of online biographies contain sentences such as "Presley was born in Tupelo, Mississippi". By the stylistic conventions of such web sites, there is little variation in these formulations. Therefore, a regex pattern like *P * born in * C* with *P* bound to a person entity and *C* to a city can robustly extract the birth places of many people. Analogously, a pattern like

```
P .* (received|won) .* (award(s)?)? .* A
```

applied to single sentences (with `(...)?` denoting optional occurrences) can extract outputs such as

```
(Bob Dylan, hasWon, Grammy Award)
(Bob Dylan, hasWon, Academy Award)
(Elvis Presley, hasWon, Grammy Award)
```

Obviously, there are many other ways of phrasing statements about someone winning an award (e.g., "awarded with", "honored by"). Manually specifying all of these would be a bottleneck; so we will discuss how to learn patterns based on distant supervision in Section 6.2.2. Nevertheless, the effort of identifying a few widely used patterns is modest and goes a long way in "picking low-hanging fruit".

Simple regex patterns with surface tokens and wildcards, like the ones shown above, often face a dilemma of either being too specific or too liberal, thus sacrificing either recall or precision. For example, the pattern *P played his X* with *P* and *X* matching a musician and a musical instrument, respectively, can capture only a subset of male guitarists, drummers, etc. Moreover, it misses out on more elaborate phrases such as "Jimmy Page played his bowed guitar". By employing NLP tools for first creating word-level annotations like *lemmatization* and *POS tags* (see Section 3.2), more expressive regex patterns can be specified, for example

```
P play $PPZ ($ADJ)? X
```

where "play" is the lemma for "plays", "played" etc. and `$PPZ` and `$ADJ` are the tags for possessive personal pronouns and adjectives, respectively. This pattern would also capture a triple ⟨`PJ Harvey, playsInstrument, guitar`⟩ from the sentence "PJ Harvey plays her grungy guitar". Further generalization could consider pre-processing sentences by *dependency parsing*, so as to capture arguments that are distant in the token sequence of surface text but close when understanding the grammatical structure. Figure 6.2 shows examples where the parsing reveals short-distance paths, highlighted in blue, between arguments for extracting instances of `playsInstrument`. The third example may fail in practice, as the path between musician and instrument is not sufficiently short. However, by additionally running *coreference resolution* (see Section 5.1), the word "himself" can be linked back to "Bob Dylan", thus shortening the path.

Note, though, that the extra effort of dependency parsing or coreference resolution is worthwhile only for sufficiently frequent patterns and properties that cannot be harvested by easier means. Moreover, parsing may fail on inputs with ungrammatical sentences, like in social media.

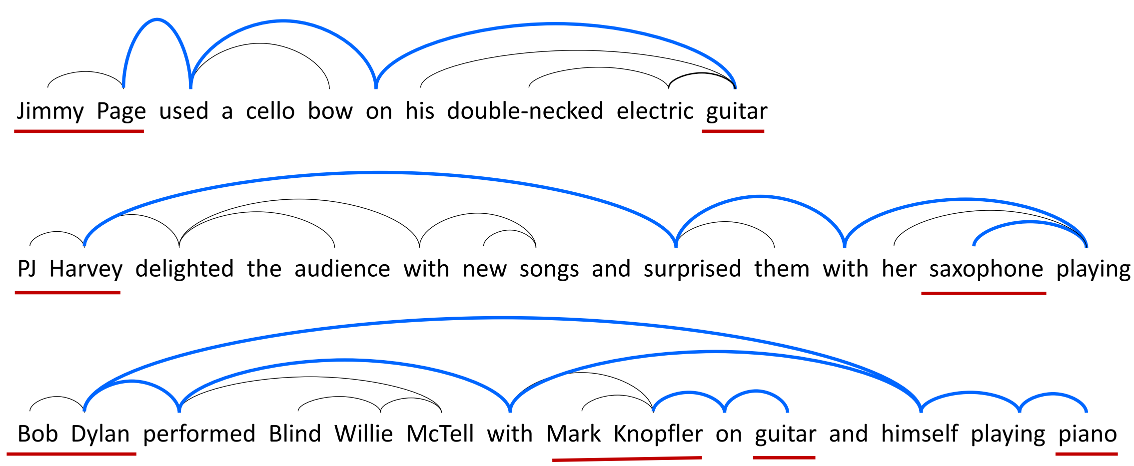


Figure 6.2: Examples for Relation Extraction based on Dependency Parsing Paths

Patterns are also frequent in headings of lists, including Wikipedia categories. The English edition of Wikipedia contains more than a million **categories and lists** with informative names such as "list of German scientists", "list of French philosophers", "Chinese businesswomen" or "astronauts by nationality". As discussed in Section 3.2 we use such cues for inferring semantic types, but we can also exploit them for deriving statements for specific properties like `hasProfession`, `hasNationality` or `bornInCountry` and more. Especially when harvesting Wikipedia, judicious specification of patterns with frequent occurrences yield high-quality outputs at substantial scale. This has been demonstrated by the *WikiNet* project [415].

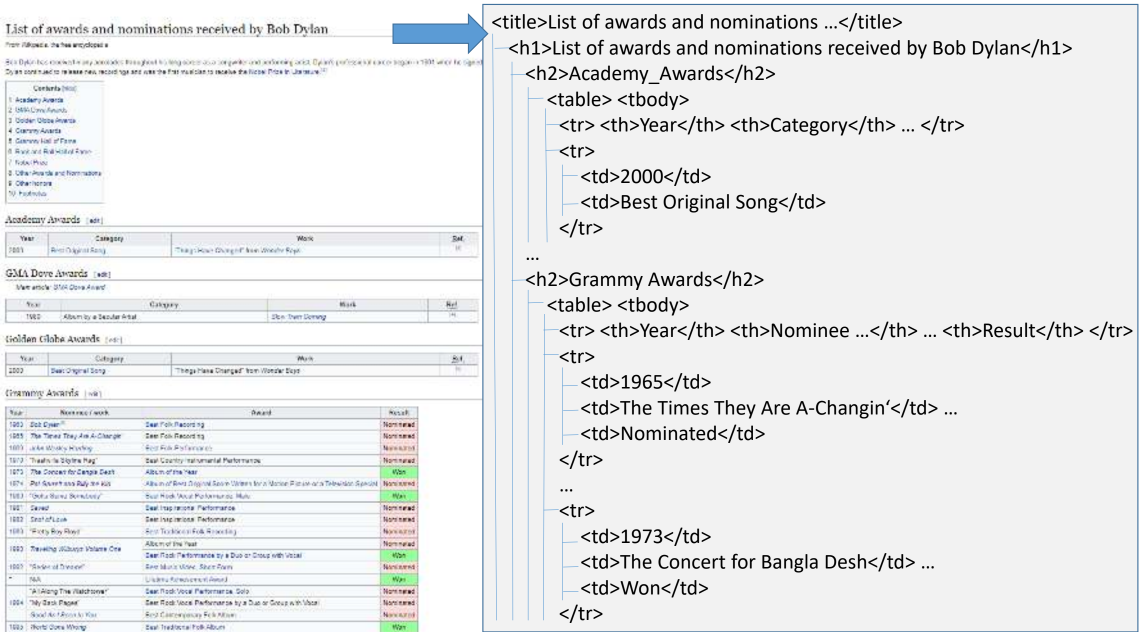


Figure 6.3: Awards List of Bob Dylan with Resulting DOM Tree

Last but not least, patterns are also found in trees, most notably, in the DOM-tree structure of HTML pages (DOM = Document Object Model, a W3C standard). Figure 6.3 shows an excerpt of a web page on Bob Dylan's awards (from Wikipedia), and outlines the DOM tree for this page. The tree has HTML tags like headings (h1, h2), table rows (tr) and table cells (td) as inner nodes and the rendered text on the leaf nodes. This tree model applies also to texts with markups in XML/XPath or the Wiki markup language (which is used by Wikipedia). Extraction patterns can be specified via rules over paths and positions of tags. For example, to identify the fact that Bob Dylan won the Academy Award in 2000 and a Grammy in 1973, the following patterns and rules can be applied:

1. locate node $\$A$ with path
   $$root \rightarrow h1 \rightarrow h2 \rightarrow \$A$$
   such that $\$A$ contains "Awards"
2. locate node $\$Y$ with path
   $$\$A \rightarrow table \rightarrow tbody \rightarrow tr \rightarrow th[k] \rightarrow \$Y$$
   such that $\$Y$ contains "Year" and $th[k]$ is the k-th occurrence of tag th as a child of the tr tag
3. locate node $\$YY$ with path
   $$\$A \rightarrow table \rightarrow tbody \rightarrow tr \rightarrow td[k] \rightarrow \$YY$$
   such that $td[k]$ is the k-th occurrence of tag td as a child of the tr tag, with k being the

same as in step 2

4. locate node $\$R$ with path

   $\$A \rightarrow table \rightarrow tbody \rightarrow tr \rightarrow th[l] \rightarrow \$R$

   such that $\$R$ contains "Result" and $th[l]$ is the l-th occurrence of tag th as a child of the tr tag
5. locate node $\$RR$ with path

   $\$A \rightarrow table \rightarrow tbody \rightarrow tr \rightarrow td[l] \rightarrow \$RR$

   such that $td[l]$ is the l-th occurrence of tag td as a child of the tr tag, with l being the same as in step 4
6. if no $\$R$ node found then output $\$A$ and $\$YY$
7. if $\$R$ found and $\$RR$ contains "Won" then output $\$A$ and $\$YY$

We give this procedurally flavored description for ease of explanation. There are formal languages and tools for expressing this extraction workflow in concise expressions, generalizing regex from strings to trees [511]. We will discuss advanced methods for extracting properties from such trees in Section 6.3.

6.2.2 Distantly Supervised Pattern Learning

6.2.2.1 Statement-Pattern Duality

Manually specified patterns go a long way, but are limited in scope, at least if we aim for high recall. Therefore, analogously to Chapter 4 on discovering entity-type pairs, we now discuss methods for automatically learning patterns. The key principle of statement-pattern duality, introduced in Section 4.3, applies to the machinery for attributes and relations as well. This was first formulated by [58] and refined in different contexts by [3, 478, 153]. We briefly recap this fundamental insight, generalizing it to triples about properties now:

Statement-Pattern Duality

When correct statements about `P(S,O)` for property `P` frequently co-occur with textual pattern p (in snippets that mention `S` and `O`), then p is likely a good pattern for `P`.

Conversely, when snippets with two arguments `S` and `O` also contain a good pattern p for property `P`, then the statement `P(S,O)` is likely correct.

Following the same rationale as in Section 4.3, this suggests an approach by **seed-based pattern learning and statement extraction** where we start with a set T of correct statements $\{P(S_1, O_1), P(S_2, O_2), \dots\}$ for property `P` (and optionally hand-crafted patterns P), and then iterate the following two steps:

1. **Pattern Discovery:**

   find occurrences of (S_i, O_i) pairs from T in a web corpus, and identify new patterns p_j

Round	Seeds	Patterns	New Statements
1	(Dylan, Blowin) (Dylan, Knockin)	$\$X$ wrote the song $\$Y$ $\$X$ wrote * including $\$Y$	(Cohen, Hallelujah) (Lennon, Imagine)
2		$\$X$'s masterpieces include $\$Y$ $\$Y$ performed by $\$X$	(Morricone, Ecstasy) (Poe, Tell-Tale Heart) (Hardy, Mon Amie)
3		$\$X$ * cover version of $\$Y$ $\$X$ * composer of $\$Y$	(Bono, Hallelujah) (Beethoven, Elise)
...	...	...	...

Table 6.1: Toy example for Seed-based Pattern Learning and Property Extraction

that co-occur with these pairs with high frequency (and other statistical measures), and add p_j to P;

2. **Statement Expansion:**
find snippets that contain i) mentions of new (S_k, O_k) pairs with proper types and ii) a good pattern in P, and add new statements $\mathtt{P}(\mathtt{S_k}, \mathtt{O_k})$ to T.

Note that we assume that entity discovery and canonicalization is performed on all input snippets, using methods from Chapters 4 and 5. So we can leverage the pre-existing KB of entities and types, and can check that the spotted entity pairs have the right type signature for the property of interest. We greatly benefit from our overriding approach of first acquiring and populating the entity-type backbone before embarking on attributes and relationships.

Table 6.1 shows a toy example for the target property

`composed: musician × song`,

using two Dylan songs as seeds. We shorten entity names for ease of reading.

The example shows that the output contains good as well as bad patterns, and some mixed blessings such as "masterpieces include". Some of the resulting false positives among the extracted statements can be detected and eliminated by type checking (e.g., Poe and his stories). Generally, aggressive pattern learning can boost the recall of the KB, but comes with high risk of degrading precision. There are two major ways of mitigating these risks:

- Corroborate candidates for new statements based on their spottings in different sources, using *statistical measures of confidence*.
- Employ *consistency constraints* to reason on the validity of candidate statements, to prune false positives.

We will elaborate on the second approach in the chapter on *KB Curation*, specifically in Section 8.5. Here we focus on confidence-driven weighting and pruning.

6.2.2.2 Quality Measures

The measures of **support**, **confidence** and **diversity** introduced in Section 4.3.2, on pattern learning for entity-type pairs, carry over to property extraction. They solely refer to patterns and statements, so that we can directly compute them for each property of interest. For easier reading, we give the definition of confidence again, in generalized form:

Given positive seeds S_0 and negative seeds $\overline{S}_0$, the **confidence of pattern** p, $conf(p)$, is the ratio of positive occurrences to occurrences with either positive or negative seeds:

$$conf(p) = \frac{\sum_{x \in S_0} freq(p,x)}{\sum_{x \in S_0} freq(p,x) \ + \ \sum_{x \in \overline{S}_0} freq(p,x)}$$

The **confidence of statement** x is the normalized aggregate frequency of co-occurring with good patterns, weighted by the confidence of these patterns:

$$conf(x) = \frac{\sum_{p \in P} freq(x,p) \cdot conf(p)}{\sum_{q} freq(x,q)}$$

where $\sum_q$ ranges over all patterns.

This way, we can score and rank both patterns and statements, and then pick thresholds for selecting the output depending on whether we prioritize precision or recall for the KB population. Moreover, we can integrate quality measures into the iterative algorithm for learning patterns and extracting statements, along the same lines that we explained in Section 4.3.2. The Extended Algorithm for Seed-based Pattern Learning applies appropriate weighting and pruning after each round, for quality control.

Just like explicitly specified patterns discussed in Section 6.2.1, learned patterns are not limited to consecutive strings in surface text. We already used wildcards in our examples, and we can also learn patterns that have POS tags as placeholders, or refer to paths in dependency parsing trees or DOM trees (see, e.g., [153, 65, 560].

6.2.2.3 Scale and Scope

Distant Supervision by Seeds from Prior KB:
The outlined methodology for pattern learning and statement harvesting relies on seed statements. These can be manually compiled, say 10 to 100 for each property of interest. But there is a more intriguing and larger-scale approach: *distantly supervised learning* from an existing high-quality KB (see, e.g., [389, 566]):

1. Build a high-quality KB by using conservative techniques with hand-crafted patterns applied to premium sources that have structurally and stylistically consistent pages. Give priority to precision, disregard recall or consider it secondary. For example, harvesting Wikipedia infoboxes falls under this regime.
2. Treat all statements of this prior KB as correct seeds for pattern learning and statement expansion. If needed, generate negative seeds (i.e., statements that do no hold) by re-combining S and O arguments that violate hard constraints (e.g., wrong birthplaces for people for whom the KB already has the correct birthplace).

Note that this method does not need any labeling of patterns or features for training, hence the name *distant* supervision.

It is straightforward to scale the method to very large inputs, as we can partition the input pages and process them in parallel (in addition to handling the properties independently). When run over multiple rounds, there is a need for handshakes regarding the statistical weights, but this is easy to implement. This makes the method perfectly amenable to Map-Reduce or other kinds of bulk-synchronous distributed computation (see, e.g., [412]).

Correlated Properties with Identical Type Signatures:
When applying seed-based learning to multiple properties, a difficult case arises whenever two properties have the same type signature and their instances are correlated. For example, `bornIn` and `diedIn` are both of type `person` $\times$ `city`, and most people die in their birth place, simply because they are born and spend their lives in large cities. In pattern learning, there is a high risk to confuse these two relations.

Another notorious example is:

`locatedIn: city` $\times$ `country` vs. `capitalOf: city` $\times$ `country`.

If we learn for the latter with prominent entity pairs as seeds, such as

`(Paris,France),(Berlin,Germany),(Tokyo,Japan), ...`

we will acquire misleading patterns such as

$\$X \ * \ \textit{largest city of} \ \$Y$.

Even if the seed set contains some cases where this does not hold, such as `(Canberra,Australia)`, the majority of seeds would still comply with the pattern. This is where **negative seeds** can play a significant role. Explicitly stating that `(Sydney,Australia)` and `(Toronto,Canada)` are *not* in the `capitalOf` relation carries much higher weight and can steer the pattern learner so as to discriminate the two confusable relations.

6.2.2.4 Feature-based Classifiers

Instead of learning patterns and then applying patterns to extract statements, we can alternatively use the features that accompany seeds and patterns to *train a classifier* that accepts or rejects new candidate statements given their features. For training, we i) spot the S and O arguments of seeds in pages and ii) observe features from local contexts such as:

- words, POS tags, n-grams, NER tags etc. to the left of S,
- words, POS tags, n-grams, NER tags etc. to the right of O,
- words, POS tags, n-grams, NER tags etc. between S and O
- words, n-grams, tags etc. in the root-to-S and root-to-O paths in DOM trees, tables, lists etc.,
- and further features that can be observed within local proximity.

When the classifier needs negative training samples, too, we use again the technique mentioned above: generate incorrect statements by replacing S and O in a correct statement with an alternative argument that is known (or at least very likely) to be incorrect.

This approach to distantly supervised property extraction has been pursued by [389] and [242], with the latter training property-specific CRF models for hundreds of properties, with little effort regarding the compilation of seeds. Advances on the TAC competition on *Knowledge Base Population (TAC-KBP)* have refined and extended this line of methods (see, e.g., [568, 265]). A typical task for these approaches is to populate a set of attributes and relations for entities of type `person` and `organization`, with properties typically found in Wikipedia infoboxes (e.g., date of birth, country of birth, city of birth, cause of death, religion, spouses, children etc. for `person`) – the sweet spot for distant supervision.

Another line of feature-based learning combines relation classification with the detection of entity mentions, into a *latent topic model* (e.g., [642]) or a **collective CRF** or other kind of probabilistic graphical model (aka. factor graph). This direction for joint inference was started by [499] and [491, 643], using the Freebase KB for distant supervision, and [673, 667, 672] for the *EntityCube/Renlifang* project [425]. Further advances were made, for example, by [332, 488, 470]. These joint models are elegant and powerful, but so far, they have not been able to demonstrate extraction accuracy at a level that allows direct import of statements into a high-quality knowledge base.

A key issue to consider here is that stand-alone entity discovery and linking, as presented in Chapters 4 and 5, have matured and become so good that it makes sense to run these steps upfront before embarking on the task of property extraction. This is an engineering argument for quality assurance, suggesting to decompose the mission of KB construction into modules that are easier to customize, optimize and debug to the specific goals of the downstream application.

Further Techniques from NLP:
When features are derived from dependency parsing structures, rather than encoding them as explicit features, a useful technique is to integrate **tree kernels** into the classifier (e.g., using SVM). This way, similarity comparisons between parsing structures are carried out only on demand. These techniques have been invented by [105, 66, 65, 400, 401] and further advanced for practical usage, for example, by [670, 584, 349].

An NLP task highly related to property extraction is **Semantic Role Labeling (SRL)** where a set of property types for an entity is specified, called a *frame* with *slots* to be filled [185, 438]. Then, given a sentence or text passage, the goal is to infer the property values for the entity. A key difference to KB population is that the input is a short text given upfront and we have the prior knowledge that the sentence and the target frame are about one and only one context (e.g., describing an event of a certain type with a fitting frame type). This is in contrast to large-scale knowledge harvesting which operates on a diverse corpus and needs to consider many targets. State-of-the-art SRL methods used to be based on deep syntactic analysis and constraint-based reasoning; see, for example, [465]. More recently, supervised end-to-end learning has been brought forward for SRL, using LSTMs and other neural networks [220, 167]. These methods are powerful, but how to utilize them for large-scale KB construction remains an open issue.

6.2.2.5 Source Discovery

Information extraction (IE) as a methodology typically assumed that it would operate on a given piece of web content. This mindset is exemplified in popular competitions and benchmarks like TAC (Text Analysis Conference), ACE (Automatic Content Extraction), SemEval (co-located with premier NLP conferences) and Semantic Web Challenges. In constructing large KBs, we have an additional degree of freedom, though, by judiciously picking the sources from which we want to extract properties. While Wikipedia is a universally good choice, KBs may need to tap additional sources for in-depth knowledge, especially when covering vertical domains such as health, food or music. Several criteria for **source quality** are important:

- **Trustworthiness:** The source has correct information about entities and their properties of interest.
- **Coverage:** The source covers many entities and many relevant properties, for the same type of entities (e.g., medical drugs or songs) or vertical domain (e.g., health or music).
- **Freshness:** The source has (nearly) up-to-date information.
- **Tractability:** The structural and stylistic conventions of the source are favorable for automated extraction, featuring, for example, crisp wording and lists, as opposed to, say, sophisticated essays.

The repertoire for identifying good sources is broad, and appropriate methods highly depend on the topical domain and the nature of web content to be tackled. Although it is tempting to think about universal *machine reading* across the entire Web [152], viable engineering needs to carefully prioritize. The following are some of the major choices:

- **Web directories:** Earlier, the Web offered good directories, such as http://dmoz.org and the original Yahoo! directory, making it easy to identify (some of) the best web sites for a category of interest. These directories are mostly defunct or frozen now, but there are still some domain-specific sites, such as http://wikivoyage.org on travel destinations, and commercial sites on hotels, restaurants etc.
- **Topical portals:** For some vertical domains, highly authoritative "one-stop" portals have emerged. These are easy to find via search engines and a bit of browsing. Examples include http://mayoclinic.org for health, http://imdb.com and http://rottentomatoes.com about movies, or http://secondhandsongs.com for cover versions of songs (to give a highly specialized case).
- **Focused crawling:** If we aim for long-tail entities or properties that are rarely featured in web sites, we may have to explore a larger fraction of the Web. The strategy here is *focused* crawling [78] where we start with a few seed pages that are known to be good, and then follow links based on a combination of classifying sites as topically relevant and scoring them as authoritative. In addition to using hyperlinks, a technique for discovering new sites is to use informative phrases from the visited pages as queries submitted to search engines. Techniques for focused crawling are described in [77, 377, 137, 539, 33, 598].

For each identified source of potential interest, the quality measures outlined above should be assessed, either via meta-sites or by sampling some of the source's pages and analyzing them [613].

For *trustworthiness*, PageRank-style metrics based on random walks have been investigated in great detail (e.g., the Eigentrust method [275]). Today, however, traffic-based measures are more informative, most notably, the Alexa rank of a site (https://www.alexa.com/). Mentioning a site as an external source in a Wikipedia page can also be seen as an endorsement of authority.

Freshness can be judged by metadata or, more reliably, by sampling pages as to whether they contain known statements of recent origin (e.g., the latest songs of popular artists).

Coverage can only be estimated, without already running the full extraction machinery on the entire site. This is best done by sampling, using seeds from the KB. However, when going for long-tail entities and infrequently mentioned properties, this may be treacherous and infeasible. Estimating the coverage, or completeness, of web sources is an open challenge. We will discuss this further in Chapter 8, specifically Section 8.1.

Finally, assessing the *tractability* of a source is a tricky issue, posing major difficulties. A pragmatic approach is to sample pages from a site, run extractions, and then manually inspect and assess the output (see also Chapter 8).

6.3 Extraction from Semi-Structured Contents

We outlined the basic approach of using path patterns in trees in Subsection 6.2.1.5. This applies to DOM trees of HTML pages as well as Web tables and lists. For deploying these techniques at Web scale, we face several bottlenecks. Suppose we tackle a single web site, such as imdb.com on movies or secondhandsongs.com on cover versions of songs. The following issues need to be addressed.

Richness of Pages:
Individual pages can be very rich, if not verbose, showing many different "sub-pages", and it is all but easy to identify the components that contain statements of interest for KB construction. This task has been addressed by techniques for page segmentation based on structural and visual-layout features (e.g., [70, 674]). For major portals such as imdb, hand-crafted rules can do this job. For tapping long-tail sites, though, feature-based learning is the method of choice.

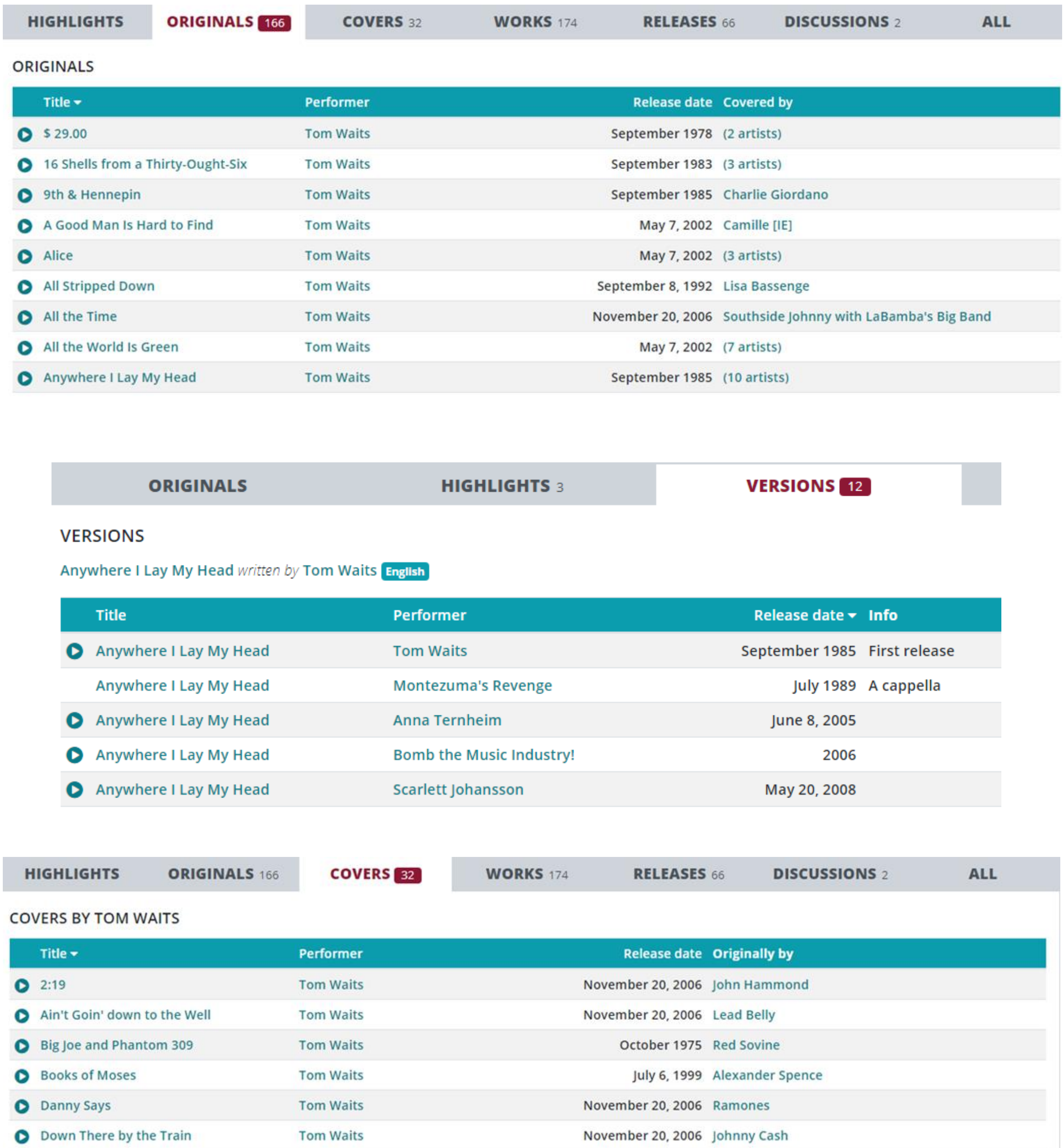
Diversity across Pages:
Even if all pages of the web site are generated from a back-end database, they cover different pieces and aspects of the site content. So there is no single path pattern that can be applied to all pages uniformly. Figure 6.4 shows an example: three different views of secondhandsongs.com featuring i) original songs of an artist, ii) cover versions of this artist's songs by other artists, iii) songs by other artists covered by the artist himself. Each of the three is instantiated for all artists, and we need three separate patterns to harvest the site. The problem is that, upfront, without manual inspection and analysis, we do not know how many different cases we need to handle. Therefore, techniques have been devised to *discover templates* among the pages of the site, based on clustering with tree-alignment and other similarity models (e.g., [668, 194]).

Cost of Labeling Training Samples:
The biggest potential showstopper is that each site and relevant template requires at least one annotated page, so that patterns and extractors can be properly induced. These costs may even arise repeatedly for the same site, namely, whenever the site owner decides to change the content slicing and layout of its pages.

The cost concern has been mitigated, to some extent, by

- visual tools for annotation (e.g., [162, 209]) and by
- hierarchical organization of patterns into *libraries* (e.g., [667, 194, 210]) for easy re-use and light-weight adaptation to different templates or even across sites.

Nevertheless, for harvesting the long tail of smaller sites within a vertical domain, it would



HIGHLIGHTS | ORIGINALS 166 | COVERS 32 | WORKS 174 | RELEASES 66 | DISCUSSIONS 2 | ALL

ORIGINALS

	Title	Performer	Release date	Covered by
▶	$ 29.00	Tom Waits	September 1978	(2 artists)
▶	16 Shells from a Thirty-Ought-Six	Tom Waits	September 1983	(3 artists)
▶	9th & Hennepin	Tom Waits	September 1985	Charlie Giordano
▶	A Good Man Is Hard to Find	Tom Waits	May 7, 2002	Camille [IE]
▶	Alice	Tom Waits	May 7, 2002	(3 artists)
▶	All Stripped Down	Tom Waits	September 8, 1992	Lisa Bassenge
▶	All the Time	Tom Waits	November 20, 2006	Southside Johnny with LaBamba's Big Band
▶	All the World Is Green	Tom Waits	May 7, 2002	(7 artists)
▶	Anywhere I Lay My Head	Tom Waits	September 1985	(10 artists)

ORIGINALS | HIGHLIGHTS 3 | VERSIONS 12

VERSIONS

Anywhere I Lay My Head *written by* Tom Waits English

	Title	Performer	Release date	Info
▶	Anywhere I Lay My Head	Tom Waits	September 1985	First release
	Anywhere I Lay My Head	Montezuma's Revenge	July 1989	A cappella
▶	Anywhere I Lay My Head	Anna Ternheim	June 8, 2005	
▶	Anywhere I Lay My Head	Bomb the Music Industry!	2006	
▶	Anywhere I Lay My Head	Scarlett Johansson	May 20, 2008	

HIGHLIGHTS | ORIGINALS 166 | COVERS 32 | WORKS 174 | RELEASES 66 | DISCUSSIONS 2 | ALL

COVERS BY TOM WAITS

	Title	Performer	Release date	Originally by
▶	2:19	Tom Waits	November 20, 2006	John Hammond
▶	Ain't Goin' down to the Well	Tom Waits	November 20, 2006	Lead Belly
▶	Big Joe and Phantom 309	Tom Waits	October 1975	Red Sovine
▶	Books of Moses	Tom Waits	July 6, 1999	Alexander Spence
▶	Danny Says	Tom Waits	November 20, 2006	Ramones
▶	Down There by the Train	Tom Waits	November 20, 2006	Johnny Cash

Figure 6.4: Examples for Three Templates on Original Songs and Cover Versions (Source: https://secondhandsongs.com/)

be desirable to completely remove the dependency on manual labeling. Next, we describe a recent approach to this goal.

6.3.1 Distantly Supervised Extraction from DOM Trees

Like with distant supervision for extraction from text, we start with seed pairs (S, O) of entities for a given property P and make two key assumptions. First, it is assumed that the web site under consideration contains many *entity detail pages* where most information on the page refers to the same entity, for example, the IMDB page on a movie or the Goodreads page on a book. Second, when processing a page about a seed entity S, we assume that the property $P(S, O)$ occurs on that page with high probability. In other words, a significant fraction of the entity detail pages contain seed statements. This is key to the viability of distant supervision. We then learn extraction patterns from these seed-matching pages, and apply them to all other pages of the web site to acquire new statements.

This high-level approach needs to overcome several difficulties, and this leads to the following processing stages, following the *Ceres* method of [354].

Distantly Supervised Method for Property Extraction from DOM Trees of Entity Detail Pages:

Input: set of entity detail pages x,
seed statements $P(S, O)$,
other statements about S from the KB

Output: set of objects T such that $P(S, T)$ for some S with page x
DOM-tree paths to T for all these objects T

For all pages x:

1. Locate subject entity S in page x.
2. Locate the dominant path to S across all pages.
3. Prune doubtful entities and pages.
4. Locate the path to T for $P(S, T)$ in page x.
5. Compute the dominant path to $P(S, T)$ across all pages.

Locating Subject Entity in Page:

There are easy cases where the URL, page title or top-level headings give proper cues. However, in cases like Figure 6.4, it is not obvious where the central entity is located for the target property `covered`: $Musician \times Musician$. In fact this varies across the three pages: `Tom Waits` in the *Performer* column for *Originals*, `Tom Waits` next to the text "$\langle song \rangle$ written by" below the heading for *Versions*, and `Tom Waits` in the header *Covers By Tom Waits* for the third page. Note that the third page can be interpreted as a source for extracting the inverse relation of `covered` so that Tom Waits is still the S entity.

To identify the central entity on each page, we first spot all S entities in the seed set for property P. For each of these candidates e (e.g., all musicians on the page) we look up its related object set $Obj(e)$ in the KB (i.e., all O entities that have some relation with the e musician), and compute *overlap* measures, like Jaccard coefficients, of $Obj(e)$ and all entities on the page. The e candidate with the highest overlap score is considered the winner. For example, for the first page of Figure 6.4, if half of the shown songs are listed in the KB as Tom Waits songs, this is high indication that we have identified the Tom Waits detail page in the web site.

Dominant Path Across Pages:
The previous step may still yield too many false positives. By assuming consistent patterns across all proper pages with entity detail, we can compute *global* evidence for the preliminary choices. Given a set of pairs (x, e) with page x being the identified candidate for the detail page on entity e, we derive the DOM-tree paths for each pair, resulting in a pool of path patterns. The most frequent pattern is considered as the proper one. Optionally, we may relax the pattern a bit, allowing for minor variations, or we can pick multiple patterns based on a frequency threshold.

Pruning Entities and Pages:
Having identified the dominant path(s) allows us to eliminate all entities and pages that do not exhibit a strong pattern. Additional heuristics can be used to further prune doubtful pages. For example, pages x that contain only very few entities related (by the KB) to the candidate entity e may be discarded, too. Likewise, if the same entity e is identified as the central entity for a large number of pages, this could be an erroneous case (e.g., due to an entity linking error) to be pruned. The implementation for this prioritization of precision requires setting threshold parameters.

Locating Paths to Objects:
With the consolidated set of proper (x, e) pairs for seed subjects, we can locate the objects O from seeds $P(e, O)$ in page x. This may lead to multiple matches of the same object in the same page, though, posing ambiguity as to whether some object mentions may refer to entities other than e (even if x is the true detail page for e) or refer to properties other than P. This is not the case in Figure 6.4, but could easily happen with these kinds of pages. Consider the third page about cover songs performed by Tom Waits. If Waits had covered multiple songs originally by Johnny Cash, we may consider the object match `Johnny Cash` as doubtful (although it is correct in this case). If the page were extended by further columns about the *singer* of the original song – which could be different from the song *writer* –, we could accidentally match a seed object in the singer column and thus pick a wrong path pattern. In this example, such cases would be rare, as song writers are often also the original singers, but the risk of spurious paths can be considerable in other settings.

To mitigate these risks, one can simply prune object matches with multiple occurrences in the same page. This sacrifices recall for the benefit of higher precision. An alternative is to check for additional evidence that the located o in page x is indeed the match for seed statement $P(e, o)$, using other objects for the same e and P in the KB. If the other objects (mostly) share the same DOM-tree path prefix as o, then all these siblings are likely in the same relation P. For example, if the KB gives us a set of song writers covered by Tom Waits, most of these should be in the same column *originally by* and only some may occur in other columns as well. The path prefix to the most frequent column wins.

Dominant Path Across All Pages:
Finally, for given property P, the best paths for every page can be compared to strengthen the *global* evidence. To this end, we define a similarity measure between paths, like edit distance scores, and all paths are clustered based on such a metric. Only clusters with high density and clear delineation from less focused clusters are selected, to retain a subset of high-quality DOM-tree path patterns.

The final set of path patterns are utilized by applying them to all identified entity detail pages of the entire web site. Comprehensive experiments by [354] have shown that this distant-supervision method is viable at Web-scale and yields high-quality output, while completely avoiding manual annotation of web pages.

6.3.2 Web Tables and Microformats

Web tables can be viewed as a special case of DOM trees. Nevertheless they have received special attention for knowledge extraction in the literature (e.g., [338, 69, 493, 435]). We already discussed methods for entity linking (EL) and column typing in Section 5.7.

By applying EL for tables, we identify S entities in column E, and we hypothesize that another table column C contains the entities (or attribute values) for the O argument of a KB property P. The key task then is to scrutinize whether C does indeed correspond to P. The method devised by [435] applies an ensemble of matchers to test this hypothesis. This includes computing the overlap between the already known O entities for P in the KB against the entities mentioned in column C. The column header, table caption and further context serve as cues for other matchers.

By applying this technique to a large collection of Web tables, a huge number of candidate statements are collected. As the candidates come from many different sites, they can be consolidated by similarity-based clustering to distill the candidates into the highest-evidence statements (see [435] for details).

Higher-arity Relations:
Tables naturally exhibit ternary and higher n-ary relations. As discussed in Chapter 2, some of these cannot be decomposed without losing information. Recall the example that someone wins the Nobel Prize in a certain field and year. This is a ternary relation `winsNobelPrize:` $person \times field \times year$ that cannot be properly reconstructed by having only the binary projections `winsNobelPrizeInField:` $person \times field$ and `winsNobelPrizeInYear:` $person \times year$. A number of methods have been developed in the literature for extracting higher-arity relations from both text and tables (e.g., [294, 293, 149, 319]).

One of the difficulties that goes beyond the case of binary relations is that observations in web contents are often partial: many pages mention only the field of the Nobel Prize winner while others mention only the year, and only few pages if any have all arguments in the same spot. This calls for reasoning over sets of extraction candidates. We will come back to this issue in Chapter 8, specifically, Section 8.5.

Microformat Annotations:
An increasing number of Web pages contain microformat markup embedded in HTML [46], using standards like RDFa, hcard or XML- or JSON-based vocabularities. For specific properties, such as addresses of organizations, authors of publications or product data, this can be a rich source, and there are specific techniques to tap microformat contents [649].

6.4 Neural Extraction

6.4.1 Classifiers and Taggers

The most straightforward way of harnessing deep neural learning for the extraction of properties is by viewing the task as a classification problem, or alternatively as a sequence-tagging problem.

For the *classifier*, the input is a sentence or another short passage of text, and the output is a binary decision on whether the sentence contains an entity pair or entity-value pair for a given property of interest. Often, the input already contains markup for candidate entities, and these text spans are fed into the classifier as well.

For the *tagger*, the input is the sentence or text snippet, and the output is a sequence of tags that identify the entity pair or entity-value pair if the network decides to accept the input as a positive case. The tags are pretty much the same as for neural NER, using BIO labels to identify the begin and inner part of tagged sub-sequences and O for irrelevant tokens; see Section 4.4.2. For example, for extracting the property `playsInstrument(Dylan,piano)`, the input

```
Dylan performed Blind Willie McTell on this rare occasion
with Mark Knopfler on guitar and himself playing piano.
```

would be tagged

```
B O O O O O O O O O O O O O O O O O B
```

Both classifier and tagger methods are trained by giving them sample sentences, positive as well as negative samples. The training is for a given property type; each property calls for a separate learner. In this regard, there is no difference from any other – simpler – machine-learning method, based, for example, on logistic regression or random forests. However, the neural learners do not take explicitly modeled features as input (e.g., dependency-parsing tags) but instead expect an embedding vector for each word (e.g., using word2vec), and nothing else. Some methods further consider word positions by encoding them into per-word vectors as well.

With these assumptions, the neural architectures that can be readily applied are the same as for NER, discussed in Section 4.4.2. Most notably, recurrent neural networks (RNNs), including **LSTMs**, and convolutional neural networks (CNNs) have been intensively studied in the literature for this purpose of property extraction from sentences. Early work [544, 217, 653] tackled the task of classifying coarse-grained lexical relations between pairs of common nouns or phrases (e.g., partOf, origin and destination of motion, etc.), and encoded word embeddings and positional information (i.e., relative distances between words) to this end. Later work integrated syntactic structures (e.g., dependency parsing), lexical knowledge such as hypernymy, and entity descriptions into their learned representations (e.g., [352, 188, 635, 509, 529, 342, 264, 665, 546, 95]).

More recent methods integrate property extraction with NER (e.g., [396, 32]), and some additionally incorporate coreference resolution into joint learning (e.g., [358]). The assumption that the input is a single sentence has also been relaxed in a few approaches that can cope with arbitrary text passages as input sequence (of limited length) (e.g., [446, 645, 506]). This way, properties can be detected where subject and object occur in separate sentences within close proximity. On the other hand, entity linking (EL) is usually disregarded. It is either assumed that the sub-sequences that denote entities in the input are already canonicalized, or EL is postponed and applied only to the entity mentions in the positive outputs of the neural extractor. Only few works have attempted to integrate EL into this kind of neural extractors (e.g., [585]).

6.4.2 Training by Distant Supervision

Since labeled training data is, once again, the bottleneck for supervised deep learning, training the outlined neural networks is pursued by distant supervision. To this end, statements from a prior KB can be used to gather sentences that contain a subject-object pair for a given property. For example, Wikipedia is a rich source of such sentences and even comes with partial entity markup, in the form of hyperlinks to entity articles or entity mentions that match article titles of prominent entities. Other corpora, such as news collections, could be considered, too.

As an example, assume the KB contains the pairs ⟨`Bob Dylan, guitar`⟩ and ⟨`Bob Dylan, harmonica`⟩ for the property `playsInstrument`. We may spot these pairs in the following sentences:

```
Bob Dylan sang and played acoustic guitar.
The audience cheered when Dylan took out his harmonica.
Dylan was accompanied by Knopfler on guitar.
The harmonica was played by Robertson, not Dylan.
```

For `playsInstrument`, only the first two of these four sentences are positive samples, and the other two would be misleading. However, we do not know which are the good ones upfront. This is a case for **multi-instance learning** with *uncertain labels*. The classifier has to cope with this noise in its training data, and this involves learning to (implicitly) distinguish proper positive samples (the first two in the example) from positive samples with spurious labels (the latter two).

The simplest technique for multi-instance learning for property extraction is to learn the best sentence from a given set, as part of the classifier training. However, this could overly focus on a single sentence, underutilizing the potential of the entire set. Therefore, a better approach is to learn weights that capture the relative influence of individual sentences towards the learned model. This is known as a **sentence-level attention mechanism** in the literature [652, 342, 264, 650], a component in CNNs, LSTMs and other networks for selecting or prioritizing input regions. The classifier could thus be enabled to properly distinguish between newly seen sentences such as

```
Bob Dylan performed the song with a tight rhythm on his piano.
Bob Dylan's accompanying band included an alto sax.
```

where only the former leads to a correct extraction.

Another difficulty that distantly supervised learners need to handle is that the same subject-object pair may appear in more than one relation. For example, the pair ⟨`Mark Knopfler, Bob Dylan`⟩ could be a training sample for all three of the properties `accompanied`, `has role model` and `friend of`. Moreover, several properties could even co-occur in the same sentence, such as:

```
Knopfler played guitar with Bob Dylan, his admired role model.
Robertson and his late friend Manuel were on Dylan's band.
```

In principle, state-of-the-art neural learners for property extraction are geared for all these difficulties. Moreover, they can be combined with other techniques like CRF-based inference to enforce consistent outputs (e.g., [666]). However, it is not yet clear how robust these methods perform in the presence of limited supervision and complex inputs. Recent research therefore explores also leveraging BERT embeddings (e.g., [542]), *reinforcement learning* and other methods that relax reliance on training data (e.g., [632, 567, 570]).

The recent article [208] reviews state-of-the-art methods and outlines challenges and opportunities in this line of neural extraction. Key issues identified for future research include:

- utilizing more data and background knowledge for better de-noising of training samples with distant supervision;
- improving the scope and scale of neural learners, to cope with longer inputs and for faster training;
- robustness to coping with complex inputs in sophisticated contexts (e.g., conversations and narrative texts, or with cues spread across wider distances);
- transfer learning to handle new property types and discover properties never seen before. We will elaborate on this theme of Open Information Extraction in Chapter 7.

6.4.3 Transformer Networks

Instead of viewing property extraction as a classification or tagging task, a different paradigm is to approach it as a **machine translation** problem. The input, or source language, is English sentences, and the output, or target language, is the formal language of SPO triples. This perspective has been pursued in recent literature, mostly building on neural learning with general **encoder-decoder** architectures (which subsume LSTMs and CNNs, although this is often not made explicit). The encoder learns a latent representation of the source language, and the decoder generates output from this learned representation into the target language.

Among the best performing and most popular models for this setting are **Transformer networks** [591] (see also [215, 7] for coding and illustrations). In a nutshell, a Transformer consists of a stack of encoders and, on top of this, a stack of decoders. Each of these components comprises a **self-attention** layer and a feedforward network. The self-attention mechanism allows the model to inform the latent representation of a word (as learned by the encoder) with signals about all other words in the input sequence. This way, Transformers capture "cross-talk" between words more directly than LSTMs where latent states accumulate distant-word influence. Moreover, Transformer networks are designed such that both training and application can be highly parallelized. The language embedding model BERT [118] is an example of a very powerful and popular Transformer network.

Recent works that leverage Transformers (including BERT) for neural extraction of properties include [596, 10, 610, 630, 55]. Another recent trend is to leverage neural methods for *Machine Reading Comprehension (MRC)* [473] to extract relational pairs of entity mentions from a given text passage. One way of achieving this is by casting the target property into one or more questions (e.g., "Who performed the song ...?") to be answered by the MRC model (e.g., [325, 334, 112]). These techniques build on the asset that their underlying neural networks have been trained with huge amounts of texts about many topics of our world.

6.5 Take-Home Lessons

Major observations and recommendations from this chapter are the following:

- Tapping into *premium sources* (ideally, with semi-structured cues like DOM trees, lists and tables) and using *highly accurate specified patterns* is the best for high-precision extraction, to keep the KB at near-human quality. For large-scale extraction, this approach has benefits also regarding declarative programming and optimization as well as scrutable quality assurance.
- For recall, and especially to acquire properties of *long-tail* entities or infrequently mentioned properties, pattern learning is a viable option. Still, harvesting high-quality web sites with *semi-structured content* by *distantly supervised extractors*, is the practically most viable way. For less prominent vertical domains, discovering these sources may be a challenge by itself.
- Extractors with *deep neural learning* have been greatly advanced, based on distant supervision from prior KB statements. This allows tapping textual contents on a broader scale, with great potential for higher recall at acceptable precision. However, the reliance on sufficiently large and clean training data (for distant supervision) is a potential obstacle, and needs further research.

7 Open Schema Construction

7.1 Problems and Design Space

In Chapter 6, we assumed that the property types of interest are explicitly specified. For example, for entities of type `musician`, we would gather triples for attributes and relationships like

```
bornOn, bornIn, spouses, children, citizenOf, wonAward,
creatorOf (songs), released (albums), musicalGenre,
contractWith (record label), playsInstrument
```

This approach goes a long way towards building densely populated and expressive knowledge bases. However, it is bound to be incomplete by missing out on the *"unknown unknowns"*: properties beyond the specified ones that are of potential interest but cannot be captured as we do not know about them yet. For example, suppose we want to expand the population of a KB about musicians such as Bob Dylan. Which properties should this KB cover, in addition to the ones listed above? Here are some candidates that quickly come to mind:

```
performedAt (location), performedAtAct (event),
coveredArtist (other musician), coveredByArtist,
workedWith (producer), performedWith (band or musician),
accompaniedBy (instrumentalist) duetWith (singer),
composed (song), wroteLyrics (for song),
songAbout (a person), etc. etc.
```

For the Bob Dylan example, noteworthy instances of these properties would include (with `Bob Dylan` abbreviated as `BD`):

```
< BD, performedAt, Gaslight Cafe >,
< BD, performedAtEvent, Live Aid Concert >,
< BD, coveredArtist, Johnny Cash >,
< BD, coveredByArtist, Adele >,
< BD, coveredByArtist, Jimi Hendrix >,
< BD, performedWith, Grateful Dead >,
< BD, duetWith, Patti Smith >,
< Sad Eyed Lady of the Lowlands, songAbout, Sara Lownds >,
< Hurricane, songAbout, Rubin Carter >, and many more.
```

Limitations of Hand-Crafted Schemas:
Although it is conceivable that a good team of knowledge engineers come up with all these property types, there are limitations in manual schema design [123] or ontology

engineering [551] To underline this point, consider the following examples. Schema.org (http://schema.org) is an industry standard for microformat data within web pages [193]. As of April 2020, it comprises ca. 800 entity types with a total of ca. 1300 property types. This is highly incomplete. For example, the type `CollegeOrUniversity`, despite having 67 specified properties, misses out on `numberOfStudents` or `degreesOffered`. Wikidata (http://wikidata.org) [600] builds on collaborative KB building; so one would expect better coverage from its large online community (further discussed in Section 9.4). As of April 2020, it specifies about 7000 property types, but still lacks many of the interesting ones for musicians and songs, such as `performedAt`, `coveredArtist`, `duetWith`, `songAbout` etc. Finally, for knowledge about movies, even the most authoritative IMDB website (http://imdb.com) misses many of the attributes and relations found across a variety of semi-structured sites: a study by [355] manually identified interesting properties about movies and found that IMDB comprises only about 10% of the ones jointly covered by eight domain-specific websites.

Key Issues:
The following sections address three key points about handling these "unknown unknowns":

- Given solely a corpus of documents or web pages, automatically *discover all predicates of interest*, using *Open Information Extraction*.
- Given a domain of interest, such as music or health, and an existing KB with a subset of specified properties, *discover new attributes and relationships* for existing entity types, using *distant supervision*.
- Given a set of discovered properties, with noise and redundancy, organize them into a clean system of *canonicalized attributes and relations* with clean type signatures, by means of clustering, matrix/tensor factorization and other data mining algorithms.

Note that once we have identified a new property of interest and have gathered at least a few of its subject-object pairs as instances, the task of further populating the property falls into the regime of Chapter 6, most importantly, using seed-based distant supervision.

7.2 Open Information Extraction for Predicate Discovery

To discover property types in text, without any assumptions about prior background knowledge, the natural and best approach is to exploit **syntactic patterns** in natural language. More precisely, we aim at *universal patterns*, or *"hyper-patterns"*, that capture relations and attributes of entities regardless of their types. The simplest pattern is obviously the basic and ubiquitous grammatical structure of the English language: *noun – verb – noun*, where nouns, or more generally, noun phrases, should denote entities in their roles as subject and object of a sentence. A simple example is "Bob Dylan sings Hurricane" with the verb "sings" being the newly detected property. The verbal part could also be a phrase, often of the form *verb* + *preposition*, like in sentences such as "Bob Dylan sings with Joan Baez" or "Bob Dylan performed at the Live Aid concert". The output of this approach is not just the property itself, but an entire **predicate-argument structure**. The verbal phrase becomes a (candidate for a) binary predicate in a logical sense, and its arguments are the subject and object extracted from the sentence. In full generality, there could be more than two arguments, because of adverbial modifiers (e.g., "Dylan performed Hurricane at the Rolling Thunder Revue") or verbs that require two objects (to express higher-arity predicates).

Open Information Extraction (Open IE)
is the task of extracting predicate-argument structures from natural language sentences.
Input: a propositional sentence (i.e., not a question or mere exclamation) that can be processed with generic methods for syntactic analysis (incl. POS tagging, chunking, dependency parsing, etc.).
Output: a predicate-argument tuple, as a proto-statement, where the predicate and two or more arguments take the form of short phrases extracted from input sentence.

To further illustrate the task, consider the sentences:

```
Bob Dylan sang Hurricane accompanied by guitar and violin.
In this concert Dylan sings together with Baez.
Bob Dylan performed a duet with Joan Baez.
Dylan performed the song with Joan Baez.
Bobby's and Joan's voices beautifully blend together.
Hurricane is about Rubin Carter and criticizes racism.
```

State-of-the-art Open IE methods produce the following proto-statements as output (e.g., https://nlp.stanford.edu/software/openie.html or https://demo.allennlp.org/open-information-extraction/):

```
< Bob Dylan, sang, Hurricane >
< Dylan, sings, together with Baez >
< Bob Dylan, performed, a duet >
< Dylan, performed, the song >
< Bobby's and Joan's voices, blend, [null] >
< Hurricane, is about, Rubin Carter >
< Hurricane, criticizes, racism >
```

A few remarks are in order:

- The S and O arguments are surface mentions in exactly the same form as they appear in the input sentence. There is no canonicalization yet, but an Entity Linking (EL) steps could be easily added for post-processing. Alternatively, one could first run entity recognition and EL on the input sentences as pre-processing.

- Some of the output triples appear incomplete and uninformative, missing crucial information. Open IE methods tend to add this as additional *modifier arguments*. For example, the sentence about the duet with Joan Baez would return:
  predicate = "performed", arg1 = "Bob Dylan", arg2 = "a duet", modifier = "with Joan Baez".
  So the crucial part is not lost, but it is not that easy to properly recombine arguments and modifiers. For example, for the first sentence, predicate, arg1 and arg2 are perfect, and the modifier "accompanied by guitar and violin" is indeed auxiliary information that could be disregarded for the purpose of discovering the predicate `sings` between singers and songs.
- Some sentences do not easily convey the subject-object arguments, an example being "Bobby's and Joan's voices" collapsed into the subject argument. As a result, the output for this sentence has no object (denoted by `[null]`).
- Some sentences, like the last one, contain information that is properly captured by multiple triples. This is a typically case for sentences with conjunctions, but there are other kinds of complex sentences as well.

7.2.1 Pattern-based Open IE

Until recently, most methods for Open IE crucially relied on patterns and rules [29, 627, 155, 374, 154, 101, 373] (with a minor role of learning-based components). Recent methods that build on neural learning will be discussed in Subsection 7.2.3. As an exemplary representative for pattern-based Open IE, we discuss the **ReVerb** method [155] which focuses on verb-mediated predicates and builds on a single but powerful **regular expression pattern** over POS tags, to recognize predicate candidates:

> **Regex Pattern for Open IE:**
> predicate = V | VP | VW*P
> V = verb particle? adverb?
> W = (noun | adjective | adverb | pronoun | determiner)
> P = (preposition | particle | infinitive marker)
>
> where determiners are words like "the", "a" etc., particles subsume conjunctions and auxiliary verbs, and infinitive markers capture the word "to" in constructs such as "Bob and Joan reunited to perform Farewell Angelina".

The above expression matches, for instance, *"sang"* (V), *"sang with"* (VP), or *"performed a duet with"* (VWP).

This way ReVerb gathers a pool of predicate candidates, and subsequently corroborates them into a cleaner set by the following steps:

1. Filter predicate candidates based on frequency in a large corpus, discarding rare ones.
2. For each retained predicate, consider the nearest noun phrase to the left as subject and the nearest noun phrase to the right as object.
3. Use a supervised model to run these (proto-)statement candidates through a classifier, yielding a confidence on the plausibility of the candidate triple.

For step 1, the huge diversity of predicates is reduced by removing adjectives, adverbs, and similar components, so that, for example, "performs a duet with" and "performs a beautiful duet with" are combined, and so are "sings with" and "occasionally sang with".

For step 3, a logistic regression classifier is learned over a hand-crafted set of training data, based on manually compiled features. Example features include *sentence begins with statement subject* (positive weight), *sentence is longer than 20 words* (negative weight), *last preposition in the predicate is a particular word, like "on", "of", "for" etc.* (different positive weights), *presence of noun phrases other than those for subject and object* (negative weight), and more. Training this classifier to reasonable accuracy required manual labeling of 1000 sentences [155].

7.2.2 Extensions of Pattern-based Open IE

The basic principles of ReVerb can be extended in many different ways, most notably, by incorporating richer syntactic cues from dependency parsing [29, 627, 374, 13] or from analyzing the clauses that constitute a complex sentence [101]. Another theme successfully pursued is to run a conservative method first, such as ReVerb, and use its output as pseudo-training data to bootstrap the learning of more sophisticated extractors. An example for the latter is the *OLLIE* method [374].

In the following, we discuss some of the less obvious extensions that together make Open IE a powerful tool for discovering predicates.

Non-contiguous and Out-of-order Argument Structure:
The regex-based approach is elegant, but falls short of the full complexity of natural language. For example, the following predicates comprise non-contiguous words:

- Dylan <u>was covered</u> among many others <u>by</u> Adele.
- Hurricane <u>made</u> the case of Rubin Carter widely <u>known</u>.

Similarly, statements do not necessarily follow the standard subject-verb-object order:

- When <u>winning</u> (**P**) the <u>Nobel Prize in Literature</u> (**O**), <u>Bob Dylan</u> (**S**) announced that he would not attend the award ceremony.

There are two ways to improve the coverage of Open IE: 1) by increasing the expressiveness of patterns, and 2) by using machine learning to generalize the pre-specified patterns. Examples of the former include the *WoE* system [627] the *ClausIE* tool [101] and the *Stanford Open IE* tool [13], using dependency parsing and clause structures. Learning was already considered by the first major system, *TextRunner* [29], but merely leveraged a small amount of labeled samples for bootstrapping a classifier. More recent systems such as OLLIE [374] *Stanford Open IE* [13] and *OpenIE 4.0* [373] systematically combine supervised learning with hand-crafted patterns.

To generate training data, the paradigm of **distant supervision** (see Section 6.2.2) can be leveraged. For example, the method of [627] matches Wikipedia infobox entries against sentences in the same articles, and treats the matching sentences as positive training samples. In contrast to learning extractors for a-priori known property types, this Open IE approach puts all properties together into a single pool of samples. This way, it can learn features that indicate new property types in text which are not covered in any of the infoboxes at all. We will come back to this form of seed-based supervision in Section 7.3.

Noun-mediated Properties:
In addition to expressing properties by verbs or verbal phrases, sometimes relations are also expressed in the form of modifiers in noun phrases. For example, the sentence "Grammy winner Bob Dylan also received an Oscar for ..." gives a strong cue for the property ⟨`Bob Dylan, winner of, Grammy`⟩. Wikipedia category names are a prominent case of this observation (see Section 6.2.1.5). Especially long-tail property types such as `wroteLyricsFor` benefit from tapping all available cues, hence the need for including noun phrases such as "Dylan's lyrics for Hurricane ...".

The works of [636, 436] are examples for tackling this issue. They operate by using seed facts to learn extraction patterns from noun phrases based on a variety of features, and then combine the learned classifier with additional rules.

Attribution and Factuality: Another limitation of the basic method is its ignorance of utterance context, such as attribution of claims. For example, from the sentence "Blogger Joe4Pacifism demanded that Dylan will also receive the Peace Nobel Prize", a simple pattern-based Open IE method would extract the triple ⟨`Bob Dylan, will receive, Peace Nobel Prize`⟩ – which is wrong when taken out of the original context. Open IE extensions such as OLLIE [374] handle this by adding *attribution fields* in their combination of pattern-based rules and supervised classification. This would yield the following output, denoted here in the form of nested tuples:

```
< Blogger Joe4Pacifism, demanded,
   < Bob Dylan, will receive, Peace Nobel Prize >
>
```

Another notorious case where simple Open IE often fails is the presence of negation cues in sentences. For example, the sentence "None of Dylan's songs was ever covered by Elvis Presley", a straightforward extractor would incorrectly yield ⟨`Dylan's songs, covered by,` `Elvis Presley`⟩, ignoring the crucial word “None”. *MinIE* [181] fixes this issue by adding a *polarity field* that is toggled when observing a negation cue from a list of keywords (e.g., not, none, never, etc.).

The *NestIE* method [41] and the *StuffIE* method [461] generalize these approaches by casting sentences into *nested-tuple structures* to represent facets like attribution, location, origin, destination, time, cause, consequence, etc. These facets are handcrafted based on lexical resources such as Wiktionary (https://www.wiktionary.org/), PropBank (https://propbank.github.io/) [439], FrameNet (https://framenet.icsi.berkeley.edu/) [165] and OntoNotes (https://github.com/ontonotes/) [460], reflecting especially the semantic roles of prepositions (“at”, “for”, “from”, “to” etc.) and conjunctions (“because”, “when”, “while” etc.). Semi-automatically annotated sentences are used to train a logistic regression classifier for casting complex sentences into nested tuples.

7.2.3 Neural Learning for Open IE

Distilling propositional sentences into predicate-argument structures has been pursued in computational linguistics for the task of *Semantic Role Labeling* (SRL, see Section 6.2.2.4). A typical setting is to consider a small set of *frame types* with specific slots to be filled. For example, for frame type `creates` they would include creator, created entity, components, co-participant, circumstances, means, manner, and more. An SRL method takes a sentence as input, uses a classifier to assign it to one of the available frame types (or none), and then applies a sequence tagger to fill all (or most) of the frame slots. This setup with arguments for *frame-specific roles* does not make sense for *Open* IE, since we do not want to limit ourselves to pre-specified frame types. Instead, we settle for more generic arguments, like arg0, arg1, arg2, etc. – an ordered list of arguments, typically with subject as arg0, object as arg1, and context-capturing modifiers as further arguments. This eliminates the need for a type classifier and focuses on the slot-filling tagger.

Not surprisingly, the state-of-the-art methodology for this sequence tagging task is deep neural networks, more specifically, bi-LSTM transducers and other recurrent networks (see Section 4.4.2). [553] presents a full-fledged solution that encodes sentences with word and position embeddings as initial inputs. The learned representation is decoded into a sequence of BIO tags to mark up the begin and inner part of predicate and argument spans (using O for all other, irrelevant tokens). To distinguish the predicate and the different arg-i’s, the BIO tags are separately instantiated for each of these constituents. An example of input and output is shown in Figure 7.1. The learning architecture itself pretty much follows those that have been developed for entity recognition (NER) and entity typing (see Chapter 4). Therefore, we do not elaborate on this any further.

An interesting aspect of [553] is the way that training data is compiled. This includes labeled sentences specifically for Open IE, but also a clever way of leveraging annotated

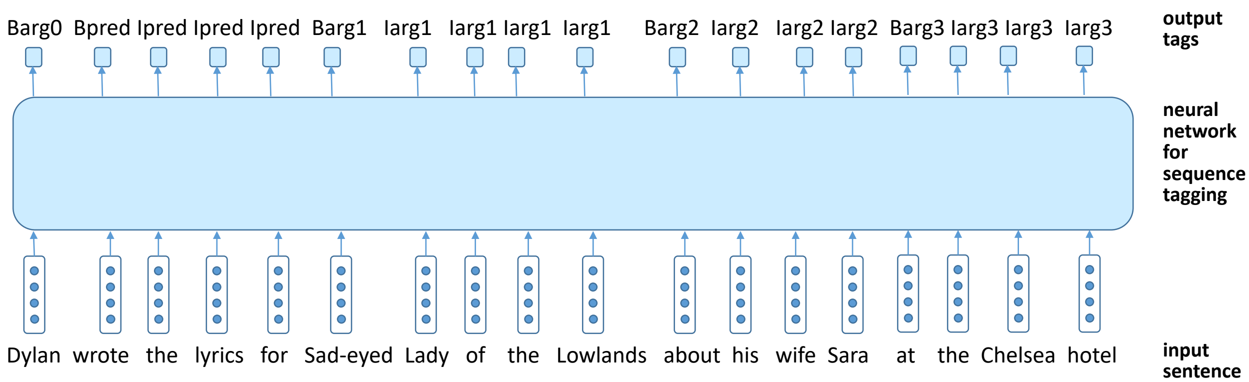


Figure 7.1: Sequence-Tagging Neural Network for Open IE

collections for QA-style machine reading comprehension [221]. The latter data comprises a large set of triples with a propositional sentence, a question that someone could ask about the sentence, and an answer that could be derived from the sentence. Questions are generated by templates and other means; they provide a user-friendly way to annotate sentences and are thus more amenable to crowdsourcing than other forms of labeling [382].

Another approach to mitigate the training bottleneck is to combine sequence-tagging methods with reinforcement learning (see, e.g., [348]).

7.3 Seed-based Discovery of Properties

In the previous section, we assumed that there is no prior knowledge base of properties: Open IE taps into any given text collection and aims at maximally broad coverage. In this section, we exploit the fact that we already have a pre-existing KB that contains many triples about a limited set of specified properties. This prior knowledge can be leveraged for distant supervision, by spotting patterns for the limited-properties statements and generalizing them into *hyper-patterns* that are applicable to discover previously unknown property types. For example, semi-structured web sites about commercial products may have many pages where salient attributes and their values are rendered in a certain style. A frequent case is list elements with the attribute name in boldface or other highlighted font followed by a colon or tab followed by the attribute value. This is a hyper-pattern.

The methods presented in this section are based on the following key intuitions:

- Property names often share similar presentation patterns, like language style (for text) or layout (for lists and table).
- Property names are much more frequent in a topically focused text corpus or semi-structured web site than their values for individual entities.

7.3.1 Distant Supervision for Open IE from Text

The **Biperpedia** project [201] took a large-scale approach to discovering new properties in web pages and query logs, by learning patterns and classifiers based on statements for pre-specified properties from a large KB. Specifically, [201] leveraged Freebase; the *WoE* system [627] pursued similar techniques at smaller scale, based on Wikipedia infoboxes. The key idea is as follows. Suppose the KB has many SPO triples for properties like

```
createdSong: musician x song
createdMovieScript: writer x movie
performedWith: musician x musician
playedOnAlbum: song x album
playedAtFestival: song x event
```

We can spot these SO pairs in a large text corpus and check that there is contextual evidence for the property P as well, by good pattern, such as:

```
song O written by S, script for O written by S,
song S played on album O, song S played live at O
```

Next, we aim to identify commonalities among patterns for multiple properties, such as

```
NN * O written by S
song S played PREP NN O
```

with noun phrases *NN* and prepositions *PREP*. These can now serve as *hyper-patterns* to be instantiated, for example, by "lyrics for O written by S" and "song S played in movie O". This is the idea to discover the previously unknown property types

```
wroteLyricsFor: artist x song
playedInMovie: song x movie
```

where the type signatures would be learned from co-occurring S-O pairs via entity linking (assuming the KB is already richly populated with entities and types).

Generally, the method runs in the following steps:

Distantly Supervised Discovery of Properties in Text:
Input: text corpus and prior KB
Output: new property types

1. Spot text snippets with occurrences of SPO triples from the KB. Use coreference resolution to capture also passages that span more than one sentence.
2. Compute frequent hyper-patterns by relaxing specific words and entities into syntactic structures (regex on part-of-speech tags, dependency-parsing tags etc.).
3. Assume that new properties are always expressed by concise nouns or noun phrases next to the parts that match S or O. Gather these noun phrases as candidates for new properties.
4. Prune noisy candidates by means of statistics and classifiers, and group candidates into synonymy-sets via co-occurrence mining or classifiers (e.g., "lyrics of O written by S" is synonymous to "text of O written by S").
5. Train a feature-based classifier to assign semantic types to the arguments of the new properties (e.g., *artist* $\times$ *song*), and run this classifier on the retained property candidates. Features include informative words in textual proximity (e.g., "rise" or "drop" as cues for datatype *number* of a discovered attribute).

The implementation of Biperpedia focused strongly on attributes rather than relations, such as

```
salesOfAlbum: album x integer
songIsAbout: song x text
```

where the O argument could be a person, location, event before entity linking or any general topic − hence the generic datatype *text*. Here, the type inference covers datatypes like numeric, date, money, text. For attributes, a fairly compact subset of hyper-patterns already gives high yield, including *NN of S is O*, *S and PREP NN O*, etc. The method itself applies to relationships between entity pairs equally well. For extracting complete SPO triples, entity linking must be run in a post-processing step or incorporated into one of the earlier steps (potentially even as pre-processing before the first step).

Tapping Query Logs:
Similar principles of distant supervision and the learning of hyper-patterns that generalize from specified properties to a-priori unknown properties, also apply to *query logs* as an input source [443, 201, 442, 347]. For example, many users ask search engines about "sales of ⟨album⟩", "lyrics writer of ⟨song⟩", "topic of ⟨song⟩", "meaning of ⟨song⟩", "what is ⟨song⟩ about?", and so on. This is a huge source for property discovery. However, it is accessible only by the search-engine providers, and it faces the challenge that user queries contain an enormous amount of odd inputs (e.g., misspelled, nonsensical or heavily biased).

7.3.2 Open IE from Semi-Structured Web Sites

Semi-structured content is a prime resource for KB construction (see Chapter 3 Section 6.3), and this applies also to open information extraction. In particular, many web sites with content generated from back-end databases organize a major part of their information into **entity detail pages**. In Section 6.3, we exploited this fact to identify pages that are (more or less) exclusively about a single entity from the prior KB. As we dealt with populating pre-specified property types in Section 6.3, this meant that we already knew S and P when tapping a web page about possible O values for an SPO triple. This greatly simplified the task, and methods like Ceres [354] showed the way to extraction with both high precision and high recall. In our current setting, with the goal of discovering new property types, we do not have the luxury anymore that we already know which P we are after, but we can still exploit that S is known for most pages from a semi-structured web site (if sites and pages are chosen carefully).

The *OpenCeres* method [355] extends Ceres to discover new properties. Earlier work along similar lines is [61] with focus on regex-based rules.

OpenCeres starts with distant supervision from the prior KB to extract and generalize patterns. This is similar to Biperpedia (see Section 7.3.1), but focuses on DOM-tree patterns. For example, many instances for a known property type could be spotted by paths in the DOM tree that end in a leaf of the form *NN:O* with a noun phrase *NN* in highlighted font followed by an entity mention O that can be linked to the KB. If *NN* does not correspond to any of the KB properties, we can consider it a newly discovered property, holding between S, the subject of the entity detail page, and O. This way, a large number of candidate properties can be gathered, from cues across all pages of the web site.

The key idea of OpenCeres lies in the subsequent *corroboration stage*, where it uses the semi-supervised method of **label propagation** [676] to clean the candidates.

Input Graph for Label Propagation over New Property Candidates: The method operates on a graph where each node is either

- a distant-training sample on a PO pair from the KB (S is assumed to be certain from the entity detail page), or
- a candidate pair of PO with P as a text phrase and O in text or entity form (i.e., before or after entity linking).

The graph connects nodes by similarity, using two kinds of features to compute edge weights:

- distance measures with regard to the DOM tree, and
- cues about visual layout such as font size, color etc.

An example graph is shown in Figure 7.2, showing PO pairs as nodes, with S being movies (not shown) from entity detail pages of the same web site. Seed nodes are in blue; edge weights are omitted for simplicity. The figures shows that entities like music pieces and musicians are connected with film directors, because they co-occur in detail pages about the respective movies. Likewise, there are movies based on books or theater plays, and these lead to edges between these literature works and the respective film director (i.e., Yimou Zhang who directed `The Banquet` and `The Flowers of War`).

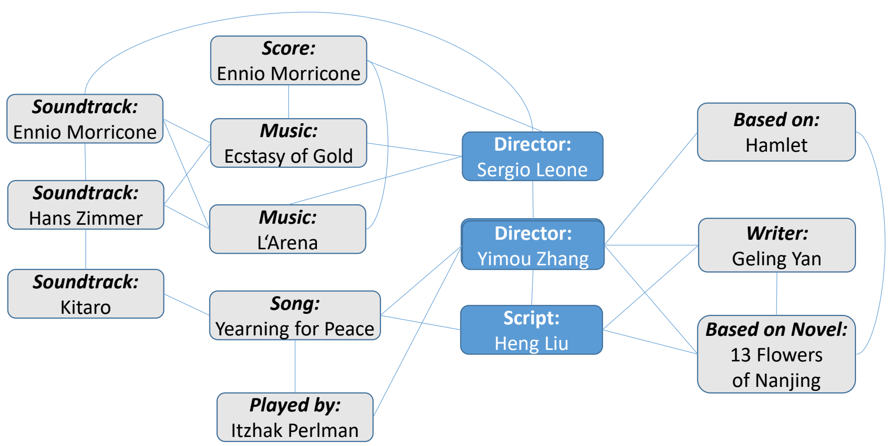


Figure 7.2: Candidate Graph for Label Propagation to Discover New Properties

Label propagation aims to assign a binary truth label to each node with a certain probabilistic confidence. To this end, it propagates the initial perfect-confidence truth of its positive seeds to neighbors in proportion to the edge eights. An underlying assumption is that the variations in truth confidence should be gradual and smooth within neighborhoods. This resembles the computation of PageRank for web-page authority, and indeed it is the same family of nicely scalable algorithms (e.g., Jacobi iteration) that can be used here. In the end, when the algorithm converges and each node has a confidence score, a threshold is applied to select the most likely proper nodes as output.

An extension called *ZeroShotCeres* [356] has further advanced this methodology by relaxing the need to have at least some seeds per web site in the distant supervision. The new method uses visual and structural cues from any domain-specific web site to learn how to tap a previously unseen web site of the same domain (e.g., movies). Technically, this is based on a *graph neural network* with graph embeddings for the visual and structural cues.

Tapping Web Tables:
The InfoGather project [637] tackled the problem of property discovery by focusing on web tables (cf. Section 6.3.2). A key idea is again using SPO triples from a prior KB for distant supervision:

1. Spot SPO in a table with SO in the same row and column header P for the column where O occurs.
2. Identify additional columns with headers Q1, Q2, ... as candidates for new properties that apply to S.
3. Aggregate these cues over all tables in the data collection, and apply statistics and probabilistic inference to compute high-confidence properties.

In addition, the discovered properties are organized into synonymy groups, using overlap measures and schema-matching techniques.

A particular challenge with web tables is that headers may be very generic and lack enough information to meaningfully identify them as informative and new. For example, column headers like *Name* or *Value* do not add any benefit to a KB schema. Even more specific headers such as *Sales* or *Growth* are doubtful, if we miss the relevant reference dimensions like currency and year. [658] extended this framework by propagating such information across tables. [253, 254] enhanced the interpretation of web tables by considering information from the text that surrounds tables in web pages, matching table cells against text phrases and, this way, picking up more context.

7.3.3 View Discovery in the KB

There are cases where the KB itself exhibits patterns that can suggest additional property types that are not explicitly specified yet. For example, suppose the KB contains properties `created`:*musician* × *song* and `performed`:*musician* × *song*. Then the composition of `created` and $\texttt{performed}^{-1}$ denotes the property `coveredBy` between musicians. Methods for rule mining (see Section 8.3.3) and path mining over knowledge graphs can automatically discover such interesting predicates. Appropriately naming them would be a modest manual effort afterwards. The *path ranking* method by [310, 311] has advanced this direction in the context of the NELL project. Note, though, that such methods can only discover what is implicitly already in the KB – there is no way to find “unknown unknowns” that have no cues at all in the KB.

7.4 Property Canonicalization

OpenIE-style methods, as discussed in the previous sections, are good at coverage, aiming to discover as many new property types as possible. However, this comes at the expense of redundancy and inconsistency. Despite some steps to clean their outputs, a method could yield two seemingly distinct properties `playedIn` and `heardIn`, both between songs and movies and both denoting the very same semantic relationship. Similarly, the output could have both `composed` and `wroteLyricsFor` between artists and songs; these are highly correlated but not semantically equivalent. To avoid all these pitfalls and arrive at unique representations of properties without redundancy and risks of inconsistencies, we need additional methods for *canonicalization* – the property-focused counterpart to what entity linking does for the property arguments S and O.

Techniques for this purpose include clustering, matrix or tensor factorization, itemset mining, and more. We start the discussion by first introducing the construction of **paraphrase dictionaries** for properties, as a building block for other techniques. This includes also inferring **type signatures** for properties, and organizing all properties into a **subsumption hierarchy**.

7.4.1 Paraphrase Dictionaries

Recall that for constructing rich type taxonomies, as a key part of the KB schema, we leveraged existing resources like WordNet and the Wikipedia category system (see Chapter3). Such assets were available for types/classes, that is, unary predicates in a logical sense, but not for properties, the binary predicates for the KB. If we had a comprehensive dictionary of binary-predicate names, along with type signatures, and each name associated with a set of synonyms, then we could use this as *complete schema* where all collected properties are positioned. In other words, there would be no more “unknown unknowns” if the dictionary were truly complete.

Needless to say that a perfect dictionary, with complete coverage and absolutely accurate synonymy-sets, is wishful thinking. Nevertheless, we can approximate this goal by constructing a reasonable dictionary and then using it as an asset in the property discovery.

The dictionary building is itself a property discovery task, though. As properties are expressed by names and phrases, such as *romance with*, *is dating* and *loving couple*, the NLP community views this as the task of **paraphrase detection**. In addition to inferring **synonymy** among such phrases (i.e., one paraphrasing the other), we also want to understand their **subsumption** relationships (i.e., one entailing the other). For example, *romance with* is a specialization of *friendship with* (at least when it is fresh), and *duet with* (between singers) is a sub-property of *performed with* (between musicians).

7.4.1.1 Property Synonymy

To infer the synonymy of two property phrases, all methods make use of the *distributional hypothesis*: the meaning of a word or phrase can by understood by observing the contexts in which it occurs.

Textual Contexts: The first methods for paraphrase detection were built directly on the distributional hypothesis applied to text corpora. The *DIRT* method (Discovering Inference Rules from Text) [339] used dependency parsing to identify subject and object of property phrases (in general text, not necessarily about KB entities). Typically, the property phrase would be a verb or verbal phrase. IDF weights (inverse document frequency) were assigned to the co-occurring subject and object, to compute an informativeness score for the property phrase. Finally, pairs of phrases could be ranked as paraphrase candidates by weighted overlap scores of their two arguments.

The approach was extended by the *OceanVerb* method [92] to consider also antonyms (i.e., opposite meanings). For example, *"in love with"* and *"enemy of"* would have high distributional similarity in DIRT, because the subject-object arguments would include common names such as Joe and Jane and even personal pronouns (as opposed to crisp entities). OceanVerb uses additional compatibility checks to infer that the example consists of opposing phrases. In principle, OceanVerb even infers semantic (meta-) relations between properties like `happened before` and `enables`. The former would hold between *"married with"* and *"divorced from"*, an example of the latter is *"composed song"* and *"played in movie"*. However, this early work was carried out at small scale, and lacked robustness.

Co-Occurrences of Entity Pairs: By restricting the subject-object arguments of property phrases to be canonicalized entities, based on a prior KB and using entity linking, the distributional-similarity signals can be made sharper. This was investigated in the *PATTY* project [414] at large scale, with extraction of candidate phrases from the full text of Wikipedia and web crawls. Each property phrase is associated with a **support set** of co-occurring entity pairs. Then, distributional similarity can be defined by overlap measures between support sets. To this end, PATTY devised techniques for **frequent itemset mining** with generalizations along two axes:

i) lifting entity pairs to type pairs and super-type pairs, and
ii) lifting words to POS tags (to replace overly specific words, e.g., a personal-pronoun tag instead of "his" or "her") including wildcards for short subsequences.

A key advantage of this approach is that the entities are typed, so that we can automatically infer type signatures for property phrases (e.g., person × person, or song × movie). In fact, the type signatures are harnessed in computing synonymy among phrases, by checking if different type signatures are compatible (e.g., musician × musician and person × person) or not (e.g., song × movie and person × movie). The PATTY method compiled a large resource of paraphrases, but still suffered from sparseness and a fair amount of noise. Most notably, its reliance on sentences that contain a property phrase and two named entities was a limiting factor in its coverage. It did not consider any coreference resolution, or other means for capturing more distant signals in text passages.

The *DEFIE* project [56] took this methodology further, specifically tapping into concise definitions, like the first sentences in Wikipedia articles or entries in Wiktionary and similar resources.

Multilingual Alignments:
Collections of multilingual documents with sentence-wise alignments, referred to as *parallel corpora* in NLP, are another asset, first considered in works by [34] and [30]. The key idea is to use pairs of sentences in language A, and their translations into language B: A1, A2, B1, B2. If A1 and A2 contain different property phrases that are translated onto the same phrase in B1 and B2, and if this happens frequently, then A1 and A2 give us candidates for synonymy. For example, consider the English-to-German translations:

A1: ...composed the soundtrack for ... → B1: ...schrieb die Filmmusik für ...
A2: ...wrote the score for ... → B2: ...schrieb die Filmmusik für ...

These are cues that "composed the soundtrack" and "wrote the score" are paraphrases of each other.

One of the largest paraphrase dictionaries, *PPDB* (Paraphrase Database) [179, 445] (http://paraphrase.org/#/download), has been constructed by similar methods from multilingual corpora with alignments. This comprises more than 100 million paraphrase pairs, with rankings and further annotations, covering both unary predicates (types/classes and WordNet-style senses) and binary predicates (relations and attributes). However, the property phrases have no type signatures attached to them.

The methodology is not limited to intellectually aligned corpora, but can be extended to leverage large collections of *machine-translated sentences*. This has been investigated by [191] using multilingual data constructed by [158] from Wikipedia sentences. By integrating entity linking (easily feasible for Wikipedia text), the method also infers type signatures for all paraphrase-sets. The resulting dictionary, called *POLY* and extending PATTY (see above), is available at https://www.mpi-inf.mpg.de/yago-naga/patty.

Note that all these methods for constructing large paraphrase dictionaries date back a number of years. With today's advances in machine learning and entity linking and the availability of much larger datasets, similar methodology re-engineered today would have potential for strongly enhanced dictionaries with both much better coverage and accuracy.

Web Tables and Query Logs: Another source for detecting distributional similarity of property phrases is web tables and query logs [222]. Both are customized cases of the **strong co-occurrence principle** introduced already in Section 4.2. For web tables, if two columns with headers P and Q in different tables contain approximately the same entities or values and both tables have high row-wise overlap of key entities (the S entities), then P and Q are good candidates for synonymy. As for query logs, the method exploits that search-engine queries that share an entity but vary in another keyword may yield highly overlapping sets of *clicked pages*. For example, the two queries "movie soundtracks by Ennio Morricone" and "film scores by Ennio Morricone" may return, to a large degree, the same good web pages that are clicked on by many users. Therefore, "soundtrack by" and "film score by" are likely synonymous. These indirect co-occurrence signals are even more pronounced for attributes such as "Paris population", "Paris inhabitants" and "how many people live in Paris".

7.4.1.2 Property Subsumption

For semantically organizing large sets of property phrases, synonymy grouping is not sufficient. In addition, we want to identify cases where one property subsumes another, this way building a taxonomic hierarchy of binary predicates. For example, `performedWith` subsumes `duetWith` between musicians, so the former should be a super-property of the latter. The PATTY, DEFIE, PPDB, and POLY projects (discussed above) addressed this issue of inferring subsumptions (or linguistic entailments) as well [414, 56, 445, 191].

The key idea in PATTY was to compare the *support sets* of different phrases (see above), computing scores for *soft inclusion* (i.e., set inclusion that tolerates a small amount of exceptions). When property phrase A has a support set that mostly includes the support set of phrase B, we can conclude that A subsumes B. This way, a large set of candidates for subsumption pairs can be derived, each with a quantitative score.

Since phrases can be generalized also at the lexico-syntactic level (e.g., replacing words by POS tags or by wildcards), this subsumption mining is carried out also for generalized phrases. For example, the phrase *VB soundtrack PREP* is more general than *wrote soundtrack for* and would thus have a larger support set, which in turn affects the mined subsumptions.

PATTY integrates all these considerations into an efficient and scalable sequence mining task, to arrive at set of candidate subsumptions. As these candidates do not necessarily form an acylic DAG, a final step breaks cycles using a minimum-cost graph cleaning technique. The taxonomic hierarchy of property paraphrases is available at https://www.mpi-inf.mpg.de/yago-naga/patty. Despite its large size, its coverage and refinement are limited, though, by the limitations of the underlying input corpora. Reconsidering the approach with modern re-engineering for Web-scale inputs and also leveraging embeddings could be a promising endeavor.

7.4.2 Clustering of Property Names

Suppose we have run an Open IE method (see Section 7.2) on a large corpus and obtained a set of proto-statements in the form of SPO triples where all three components are short text phrases. This may give high recall but comes with noise and diversity: many variants of P phrases potentially denoting the same property and ambiguous phrases for S and O. An obvious idea to clean this data and organize it towards crisper triples for KB population, KB schema construction or schema extension with new properties, is by **clustering** entire triples or P components. Several methods have been developed to this end [646, 387, 463, 175]; they differ in specifics but share the same key principles, as discussed in the following.

Consider the following example output of Open IE:

```
< Bob Dylan, sang with, Patti Smith >
< Dylan, sang with, Joan Baez >
< Bobby, duet with, Joan >
< Dylan, performed with, Baez >
< Dylan, performed with, violinist Rivera >
< Dylan, performed with, his band >
< Dylan, performed with, Hohner harmonica >
< Bobby, shot by, assassinator >
< Hurricane, is about, Rubin Carter >
< Hurricane, criticizes, racism >
< Sara, is about, Dylan's wife >
< Green Mountain, is about, civil war >
< Green Mountain, critizes, civil war >
< Masters of War, criticizes, war >
```

Pre-processing of S and O Arguments:

To have a clearer focus on canonicalizing the properties, most methods first process the S and O arguments towards a cleaner representation. An obvious approach, if a prior KB about entities and types is available, is to run *entity linking* on both S and O. If this works well, we could then group the triples by their proper S and/or O arguments. For example, the first 5 triples all have `Bob Dylan` as S argument. The sixth triple, which actually refers to `Robert Kennedy`, may produce a linking error and would then be spuriously added to the previous group. The KB could further serve to lift the entities into their most salient types, like `musician` and `song` for the S arguments. These types can further annotate the groups as features in downstream processing.

If no KB exists a priori, then the method of choice is to cluster all S arguments and all O arguments based on string similarity and distributional features like co-occurrence of P and O phrases and tokens in the underlying sentences or passages.

The result of this stage would be a set of *augmented triples* where each unique P phrase is associated with a group of S arguments and a group of O arguments (where non-entity phrases are kept as strings):

```
< {Bob Dylan}, sang with, {Patti Smith, Joan Baez} >
< {Bob Dylan}, duet with, {Joan Baez} >
< {Bob Dylan}, performed with,
  {Joan Baez, Scarlet Rivera, his band, Hohner harmonica} >
< {Bob Dylan}, shot by, {assassinator} >
< {Hurricane, Sara, Green Mountain}, is about,
  {Rubin Carter, Sara Lownds, civil war} >
< {Hurricane, Masters of War}, criticizes, {racism, civil war, war} >
```

Similarity Metrics for Clustering

The main task now is the clustering of the P phrases themselves. The crucial design decision here is to choose features and a similarity metric that guides the clustering algorithms. For this purpose, prior works investigated combinations of the following feature groups:

Features for Property Clustering

- **String similarity** of property phrases P and Q, potentially incorporating words embeddings, so that, for example, *sang with* and *performed with* would be similar.
- **Support sets overlap** where the support sets for phrase P are the groups of S and O arguments associated with P (see example above).
- **Type signature similarity** between P and Q, with the semantic types of S and O arguments derived from the support sets, such as `musician`, `singer`, `song` etc. The scores for how close two types are to each other can be based on word embeddings or on other relatedness measures derived from the KB´s taxonomy or entity-linked data collections (cf. Section 5.7).
- **Relatedness in property dictionary**, to leverage an existing dictionary such as PPDB or PATTY (see Section 7.4.1) where a subset of property synonyms or near-synonyms (e.g., "plays with", "performs with" and perhaps also "sings with") are already known, as are highly related properties in the subsumption hierarchy.
- **Context similarity** based on the sentences or passages from where the triples were extracted, again with the option of word embeddings (cf. Section 5.3.2).

Algorithmic Considerations:
The algorithm itself could be as simple as *k-means clustering*. The output for our toy example could be as follows, with k set to three:

```
cluster 1: {sang with, duet with, performed with}
cluster 2: {shot by}
cluster 3: {is about, criticizes}
```

This would be a fair set of clusters. Still it confuses duets of singers with musicians performing with each other, and it does not discriminate the different semantics of `is about` and `criticizes`. A generally critical issue is the choice of the number of clusters, k. If we had set k to 4, it may perhaps separate `is about` and `criticizes`, but the output could also get worse, for example, by combining `performed with` and `shot by` into one cluster (which could well be due to "shot by´´ also having a meaning for movies and videos).

For more robust output, one should adopt more advanced algorithms like *soft clustering* or *hierarchical clustering*. The latter has been used by prior works [646, 387, 175]. It allows picking a suitable number of clusters by principled measures like the Bayesian Information Criterion. For efficiency, the actual clustering can be preceded by simpler *blocking techniques* (cf. Section 5.2), such that the more expensive computations are run only for each block separately.

Another, more far-reaching, extension is to cluster properties P and SO argument pairs *jointly* by using *co-clustering* methods. Such approaches have been developed, for example, by [463, 437]. These methods do not perform the pre-processing for SO pairs, but instead integrate the clustering of entities and/or SO phrases with the clustering of properties. The factorization method discussed in the following section can also be seen as a kind of co-clustering. Another variation of this theme is the work of [420] where Open IE triples are first organized into a graph and then refined towards canonicalized representations by iteratively merging nodes. This approach has been further enhanced by integrating property canonicalization with entity linking [340].

7.4.3 Latent Organization of Properties

An alternative to explicit grouping of property phrases, with type signatures, synonymy and subsumption, is to organize all triples from Open IE into a latent representation, using **matrix or tensor factorization**. These approaches take their inspiration from the great success of such models for recommender systems [289, 288, 664]. A typical recommender setting is is to start with a user-item matrix whose entries capture the purchases or likes of online users regarding product items, or numeric ratings from product reviews. Naturally, this matrix is very sparse, yet it latently reflects the users' overall tastes; hence its usage to generate recommendations for new items. The input matrix is factorized into two low-rank matrices such that their product approximates the original data but in a much lower-dimensional space of rank k. One factor matrix maps users into this latent space, and the other one maps items. By re-combining user and item representations in the joint latent space, we compute how well a user may like an item that she has never seen before. This is also referred to as *matrix completion* as it fills in the initially unknown values of the sparse input matrix.

An approach to property organization along similar lines has been developed by [492, 644] under the name **Universal Schema**. The basic idea is to arrange the collected Open IE triples into a matrix and then apply low-rank factorization to map both SO pairs and P phrases into the same latent space.

> **Latent Organization of SO-P Data:**
> The input data for computing a latent representation is an $m \times n$ matrix M whose rows are SO pairs (phrases, as proxies of entity pairs) and whose columns are P phrases.
> The output is a pair of factor matrices $U : m \times k$ and $W : k \times n$ with a specified low rank k, such that their product $\tilde{M} = U \times W$ minimizes a distance norm (e.g., Frobenius norm) between M and $\tilde{M}$.

This low-rank factorization reduces the original data to its gist, capturing cross-talk and removing noise. We may think of this as a *soft clustering* technique, as it leads to representing each SO pair and each P phrase as a k-dimensional vector of soft-membership scores. The rows of U are latent vectors of SO pairs, and the columns of W are the vector for P phrases. Computing the factorization requires non-convex optimization as it typically involves additional constraints and regularizers. Therefore, gradient descent methods are the algorithms of choice.

Inference over Latent Vectors:
The latent form of Universal Schema makes it difficult to interpret for human users, including knowledge engineers. It still remains unclear how the Universal Schema data can be incorporated for KB curation and long-term maintenance. Nevertheless, there are valuable use cases for harnessing the latent data.

To determine if two P phrases denote the same property, we compare their latent vectors, for example, by the scalar product or cosine of the respective columns of W. Analogously, we can test if two SO pairs denote the same entity pairs. This is not a crisp true-or-false solution, yet it is a mechanism for interpreting the Open IE data under a *universal but soft "schema"*. Moreover, it is a building block for other downstream tasks, particularly, testing or predicting if a given property would likely hold between two entities. This resembles and is closely related to *Knowledge Graph Embeddings* [423, 611, 266, 503], to be discussed in Section 8.4.

A drawback of matrix factorization models is that they are inherently tied to the underlying input data. So any inference we perform in the latent space is, strictly speaking, valid only for the original triples and not for any newly seen data that we discover later. However, there are workarounds to this issue. Most widely used is the so-called fold-in technique: when we spot a previously unknown P phrase in a new set of triples, we organize it as if it were an $n+1^{st}$ column for the data matrix M, capturing all its co-occurrences with SO pairs. The multiplication $W \times M_{*,n+1}$ of the factor matrix W with this additional vector maps the vector into the k-dimensional space. From here, now that we have a latent vector for the new phrase, we can compute all comparisons as outlined above.

There is still a problem if the new P phrase comes with SO pairs where an S or an O are not among the entities captured in the original matrix M. The "row-less" Universal Schema method [595] adds techniques to handle property phrases whose observations include newly seen entities ("out-of-KB" entities if M is viewed as a pseudo-KB).

Extensions:
The Universal Schema method has been extended in a variety of ways. Instead of treating SO pairs as one side of an input matrix, one can arrange all data into a *third-order tensor* with modes corresponding to S, P and O phrases. This tensor is decomposed into low-rank matrices which yield the latent representations of S arguments, P phrases and O arguments (see, e.g., [147, 80]).

Another extension is to combine inputs from Open IE triples and a pre-existing KB, where the KB is confined to pre-specified properties, the triples may contain new property types unknown so far, and both inputs overlap in the entities that they cover [595]. In a similar vein, multilingual inputs can be combined into a joint matrix or tensor, for higher coverage [594]. This exploits co-occurrences of SO pairs with property phrases in different languages.

Finally, more recent works on Universal Schema have devised probabilistic and neural models for learning the latent representations of S, P and O phrases [595, 657], replacing the matrix/tensor factorization machinery. Essentially, a classifier is trained for P phrases to denote one of n existing property types or a new, previously unseen, property. The input for learning the classifier is the vectors (and additional embeddings) for the S, P and O components of triples from both prior KB and Open IE collection. The objective is to learn an "entity-neighborhood-aware" encoder of property phrases. When the trained model receives a new triple of SPO phrases, it makes the classification decision, as mentioned above. P phrases which are most likely none of the KB properties become (still noisy) candidates for extending the KB schema at a later point.

7.5 Take-Home Lessons

Major conclusions from this line of research are the following:

- Open information extraction, from any given text collection without prior KB, is a powerful machinery to *enhance recall* towards better coverage of KB construction. It is able to discover new properties, but this often comes with substantial noise, adversely affecting precision.
- Using *distant supervision* from known properties in a *prior KB* can help to improve the precision of property detection, especially when tapping semi-structured web sites.
- *Canonicalizing* the resulting SPO triples, especially the newly seen property phrases, for inclusion in a high-quality KB still poses major challenges. Methods based on distributional-similarity cues employ *clustering* or *data mining* techniques, either to construct *paraphrase dictionaries* or for cleaning of Open IE collections.
- Organizing SPO triples from Open IE in a *latent low-rank space* is an interesting alternative to explicit schemas, but latent “schemas” are not easily interpretable. The role of such approaches in KB curation and long-term maintenance is still unclear.

8 Knowledge Base Curation

Although the KB construction methods presented in the previous chapters aim for high-quality output, no KB will be anywhere near perfect. There will always be some errors that result in incorrect statements, and complete coverage of everything that the KB should have is an elusive goal. Therefore, additional **quality assurance** and **curation** is inevitable for a KB to provide, maintain and enhance its value. This aspect is of utmost importance as the KB grows and evolves over time, with deployment in applications over many years.

We start this chapter by introducing quantitative metrics for *KB quality* (Section 8.1) and the degree of *completeness* (Section 8.2). A key instrument for quality assurance is logical *constraints* and *rules* (Section 8.3), as well as their latent representations via *graph embeddings* (Section 8.4). Consistency constraints, in particular, are crucial for detecting and pruning false positives (i.e., spurious statements) and thus facilitate the *KB cleaning* process (Section 8.5). We conclude this chapter by discussing key aspects in the long-term *life-cycle* of a KB, including provenance tracking, versioning and emerging entities (Section 8.6).

8.1 KB Quality Assessment

8.1.1 Quality Metrics

The degrees of correctness and coverage of a KB are captured by the metrics of **precision** and **recall**, analogously to the evaluation of classifiers. Precision and recall matter on several levels: for a set of entities that populate a given type, for the set of types known for a given entity, for the set of property statements about a given entity, or for all entities of a given type. For all these cases, assume that the KB contains a set of statements S whose quality is to be evaluated, and a ground-truth set GT for the respective scope of interest.

Precision is the fraction of elements in S that are also in GT:

$$precision(S) = \frac{S \cap GT}{S}$$

Precision may also be referred to, in the literature, by the terms accuracy, validity or correctness (with some differences in the technical detail).

Recall is the fraction of elements in GT that are also in S:

$$recall(S) = \frac{S \cap GT}{GT}$$

Recall may also be referred to by the terms completeness or coverage.

Knowledge extraction methods, from both semi-structured and textual contents, usually yield *confidence scores* for their output statements. To compute precision and recall for confidence-annotated statements, one needs to choose a threshold above which statements are retained and below which statements are discarded. For example, for a binary classifier, we could use the odds of its scores for accepting versus rejecting a candidate statement as a confidence value. For advanced machine-learning methods, this can be much more sophisticated, though. Properly calibrating confidence scores for human interpretability poses difficulties depending on the specific choice of the learner (see, e.g., [74]). Moreover, some methods, such as MAP inference for probabilistic graphical models [283] (cf. Section 5.5.4), produce only joint scores for entire sets of statements, which cannot be easily propagated to individual statements. Computing marginal probabilities is often prohibitively expensive for such joint inference methods.

To avoid the difficult and highly application-dependent choice of an adequate threshold, a widely used approach is to compute precision and recall for a range of threshold values, and to inspect the resulting **precision-recall-tradeoff-curve**. This curve can be aggregated into a single metric as the **area under the curve (AUC)**, also known as **average precision**, or alternatively, the area under the receiver-operating-characteristic curve (ROC). Also, for each of the points where we know precision and recall, we can compute their harmonic mean, and use the best of these values as a quality metric. This is known as

$$\textbf{F1 score} = \frac{2}{precision^{-1} + recall^{-1}}$$

Besides these standard measures, further metrics of interest are the *density* and *connectivity* of the KB: the number of statements per entity and the average number of links to other entities. Also, the number of distinct property types for entities (of a given type) can be interpreted as a notion of *semantic richness*. *Freshness* with regard to capturing up-to-date real-world facts is another important dimension. More discussion on KB quality measures can be found in [444, 157, 454, 227]. For taxonomic knowledge, that is, the hypernymy graph of types (aka. classes), specific measures have been studied (e.g., [53]).

8.1.2 Quality Evaluation

The definitions of precision and recall require ground truth GT, such as all songs by Bob Dylan or all artists who covered Bob Dylan. Since obtaining comprehensive ground truth is often impossible, evaluating precision and recall usually resorts to **sampling**.

Evaluating Precision:
For precision, this entails selecting a (random) subset of elements from the KB, and for each of them, deciding whether it is correct or not. For example, we could pick a few hundred statements for a given property type uniformly at random and have them assessed [240]. More sophisticated sampling strategies for KBs are discussed in [180].

The assessment is by human judgement, which may come from crowdsourcing workers or, if needed, domain experts (e.g., for biomedical knowledge). Such gathering of ground-truth data requires care and effort in terms of annotation guidelines and quality control, especially if relying on laypeople on crowdsourcing platforms. Also, assessments often have inherent uncertainty, as the annotators may misjudge some statements depending on their personal background and (lack of) thoroughness. By evaluating different subsets of the sampled data points, the process can also serve to approximate precision-recall curves [505].

A common scheme is to have multiple annotators judge the same samples, and to consider assessments only as ground-truth if the inter-annotator agreement is sufficiently high. In this process, some annotators may be dismissed and the agreement and outcome could be based on annotator-specific confidence weights, performing a form of weighted voting (e.g., [574]). The overviews [124, 91, 8] give more background on handling crowdsourcing tasks.

Evaluating Recall:
For recall, evaluation based on sampling is more difficult, as the key point here is coverage with the goal of capturing *all* knowledge of interest. A common approach is to have human annotators hand-craft small collections of ground-truth, for example, all songs by Bob Dylan or all movies in which Elvis Presley starred. In contrast, for evaluating the quality of an automated extraction method and tool, annotators have to read an entire text or web page and mark up all statements that would ideally be captured for a KB.

Obviously, both tasks are labor-intensive and do not scale well. Therefore, a proxy technique is to evaluate a KB or extractor output relative to another pre-existing KB (or dataset) which is known to have high recall with regard to a certain scope (e.g., songs and movies of famous musicians) [444].

8.1.3 Quality Prediction

Often, the sources from which a KB is built cannot be easily characterized in terms of confidence. This holds, for example, for collaboratively created knowledge bases where many users enter and edit statements (e.g., Wikidata). For automatic extraction, on the other hand, it is often desirable to estimate the quality that a certain extractor obtains from a certain source. This is a building block for assessing the resulting KB statements, and it is also useful as an a-priori estimator before paying the cost of actually running the extractor at scale. These settings call for **predicting** the output quality based on features of the underlying sources and methods.

For the case where KB content is created by online users, three kinds of signals can be used:

1. *user features*, such as general expertise and the number of previous contributions,
2. features of the *KB subject or statement* under consideration, such as popularity or controversiality, and

3. *joint features* of the user and the subject [574, 226] (e.g., user living in a country on which she adds KB statements).

A finding in [574] is that the third group of features is most important: quality of contributions correlates more with topic-specific user expertise than with general user standing or the popularity of the topic. These findings are based on empirical data from editing the Freebase KB. On the other hand, [226] reports that for the Wikidata KB, user features, such as the number of edits and the user status, are indicative for quality. Other studies have looked at the edit history of Wikipedia (e.g., [337, 298, 249]). All the above features can contribute to counter vandalism and other kinds of quality-degrading edits.

Knowledge Fusion:
For the case of using automatic extractors, a key issue is to compare and consolidate outputs from different sources and methods, like statements obtained from DOM trees, from web tables, from query logs, and from text sources, or statements from different web sites on the same topic. By quantifying the quality of each *source/extractor combination*, their outputs can be given adequate weights in the corroboration process. The **knowledge fusion** method of [129, 131] modeled and quantified the mutual reinforcement between output quality and source/extractor quality. The output statements have higher quality if they come from a better source and extractor, and the source and extractor have higher quality if they yield more correct statements. This line of arguments is related to the principle of statement-pattern duality (see Sections 4.3.2 and 6.2.2.1) as well as to joint learning over probabilistic factor graphs. We discuss this direction further in Section 8.5. A key finding from [129, 131] is that the best quality was obtained from DOM trees (see Sections 6.3.1 and 7.3.2), whereas both query logs and web tables provided rather poor quality due to high noise.

Evidence Collection:
The knowledge fusion techniques can also be used to validate or refute KB statements collected from multiple sources. This is particularly challenging for knowledge about long-tail entities. To this end, [327] devised strategies for collecting evidence in support of candidate statements or counter-evidence for falsification. The compiled pieces of evidence are then subject to jointly assessing the quality of sources and statements, following the outlined idea of knowledge fusion.

[327] developed a complete fact-checking tool called **FACTY**, successfully used for KB curation. Section 9.5 gives more details on FACTY as part of an industrial KB infrastructure. Other fact-checking methods have been studied, for example, by [648, 333, 411, 173]; an overview on this topic is given by [335].

Predicting Recall:
Again, recall is more tricky and tedious to estimate. Prior work on text-centric query

processing [258] investigated selectivity estimators, but these do not carry over to KB quality. Viable approaches exist for special cases, though. For predicting the **recall of types**, that is, the instances of a semantic class such as folk songs or Grammy award winners, statistical methods for *species sampling* can be used [586, 359]: given a set of birdwatchers' partial counts of different species in the same habitat, what is (a good statistical estimator for) the population size for each of the species? In the KB setting, we would have to sample counts for class instances from different sources, and then estimate the total number of distinct instances. [359] shows that the growth history of the Wikidata KB can be used to derive multiple samples, and the overlap between these samples yields estimates for the size of semantic classes. Other statistical techniques along similar lines include using Benford's Law on the distribution of class cardinalities [547].

An alternative to these intrinsic predictions are extrinsic studies and coverage models on how well KBs support typical workloads of queries and questions [246, 479].

Textual Cues for Cardinalities and Recall:
Sometimes, there are ways to estimate the cardinality of a semantic class or the cardinality of the distinct objects for a given subject-property pair, even without being able to extract the actual entities. In particular, text sources often contain cues such as:

```
Bob Dylan released 51 albums
Bob Dylan's works include 39 studio and 12 live albums
On his 51st album Dylan covered Sinatra songs
Frank Sinatra and his first daughter Nancy
Frank Sinatra and his second daughter Tina
There are more than two thousand Grammy award winners
```

These sentences contain numbers, numerals (i.e., text expressions that denote numbers) and ordinal phrases (e.g., "51st", "second"). From these we can infer cardinalities or at least lower and upper bounds. Adding this knowledge, in the form of **counting quantifiers**, enhances the KB. The extraction is not straightforward, though. On one hand, there is the diversity of surface expressions; on the other hand, there are overlapping and noisy cues. For example, we must not over-count albums by adding up studio albums, live albums, albums from the 20th century and albums from the 21st century. [390] presents techniques for extraction and consolidation of such cardinalities.

Counting quantifiers can be used to assess recall by comparing a confirmed count (e.g., for the number of Bob Dylan albums) against the KB instances of the respective set. This can in turn reveal gaps in the KB, and can steer the knowledge gathering process to specific enrichment of the KB contents [391].

The textual form of expressing lists of objects for a given subject-property pair can also give cues about recall. For example, the sentence "Dylan's favorite instruments used to be guitar and harmonica" suggests that there could be more instruments, whereas the sentence

"Bob Dylan has played guitar, harmonica, banjo, piano and organ" implies high recall or even completeness. [480] presents methods for leveraging such textual cues towards recall estimation, based on linguistic theories of communication.

8.2 Knowledge Base Completeness

No knowledge base can ever be fully complete. This suggests that, unlike database, which are traditionally interpreted under a *Closed-World Assumption (CWA)*, we should treat KBs as following an *Open-World Assumption (OWA)*:

> The **Open-World Assumption (OWA)** postulates that if a statement is not in the KB, it may nevertheless be true in the real world. Its truth value is unknown. In other words, absence of evidence is not evidence of absence.

Whenever we probe a statement and the KB does not have that statement (and neither a perfectly contradictory statement), we assume that this statement may be true or may be false – we just do not know its validity. For example, even when a rich KB contains 500 songs by Bob Dylan and 1000 artists who covered him, Boolean queries such as "Has Bob Dylan written the anthem of Vanuatu?" or "Has Bob Dylan ever been covered by Woody Guthrie?" have to return the answer "maybe" (or "don't know").

This is at least in the absence of knowing a different composer of the Vanuatu anthem, and not being able to rule out a covering artist because he or she died already before Dylan was born. The answer cannot be "no" even if both statements are sort of absurd.

8.2.1 Local Completeness

Notwithstanding the general OWA, we are often able to observe and state that a KB is *locally complete*. For example, a human knowledge engineer may assert that the instances of a specific type, such as `Nobel laureates` or `Grammy winners`, are complete. Then all queries of the form "Did Elvis Presley win the Nobel Prize?" can faithfully return the answer "no".

This idea can be generalized to other localized settings, most notably, to the set of objects that would be in the range of a given subject-property pair. For example, the KB could have a complete list of Bob Dylan albums, perhaps even a complete list of his songs, but only a partial set of artists who covered him. Formally, we specify that the sets

$\{O | \langle \texttt{Bob Dylan}, \texttt{releasedAlbum}, O\}$ and

$\{O | \langle \texttt{Bob Dylan}, \texttt{composedSong}, O\}$

are complete. This helps querying under CWA semantics for the feasible queries, and it helps the KB curation process, as we know that further triples for these SO pairs that human contributors or automatic extractors may offer should not be accepted.

The principle that underlies these considerations is the following [174, 129]:

The **Local Completeness Assumption (LCA)**, aka. *Partial Completeness Assumption* or *Local Closed-World Assumption*, for entity S in the KB asserts that if, for some property P, the KB contains at least one statement $\langle S, P, O\rangle$, then it contains all statements for the same S and P. That is, there exists no object X for which $\langle S, P, X\rangle$ holds in the real world but is not captured in the KB.
For example, if we know one of a person's children, then we know all of them; S is complete with regard to P.
A generalized form of the LCA that applies to *entire classes* asserts that for all entities S of a given type T and a given property P, the set of objects for SP is complete.

The rationale for the LCA is that salient properties of important entities will be covered completely, throughout the KB maintenance process, or not at all. Hence the LCA for properties like children, types like Nobel laureates, and subjects like premier league football clubs. [482, 176, 27] conducted empirical studies on how well the LCA holds for large KBs like Wikidata. The studies did find gaps (e.g., only subsets of children known), but by and large, the LCA is often a valid assumption. More precisely, this holds for relations in the direction that is closer to being functional (i.e., with smaller range of distinct objects per subject). For example, the LCA is empirically supported for `hasCitizenship:` *person* $\times$ *country*, but not for its inverse relation `hasCitizen:` *country* $\times$ *person*.

Of course, for SP object sets that may still grow over time, the LCA has to be checked again every now and then. But there are many cases with immutable sets where a *completeness guarantee* would indeed freeze the relevant part of the KB. Examples are the parents of people, the founders of companies, or movies by directors who are already dead. Instead of the complete set being the objects for an SO pair, we can also consider the situation with a set of subjects for a given PO pair. Furthermore, the local completeness could be conditioned on an additional property, most notably, on the type of the S entities. For example, we could assert completeness of parents and children for all politicians or all EU politicians, but perhaps not for artists (who may have a less documented life – at least by the cliché).

Completeness Assertions:
Starting from seminal database work [403], proposals have been made to extend knowledge bases with *completeness statements*, to assert which parts of the data should be treated under LCA semantics. By default, all other parts would then be potentially incomplete, to be treated under OWA semantics. Formal languages and patterns for expressing such assertions have been investigated, for example, by [481, 111, 307] for relational databases and for the RDF data model.

The RDF standard itself can not express these assertions, though. Instead, we can turn to the Web Ontology Language OWL 2 [601], which has different ways of specifying local completeness. The following are OWL examples for asserting that i) there are only fourteen eight-thousander mountains, and that ii) people have at most one birthdate:

```
i) :classOfEightthousanders owl:oneOf
     ( :Everest :K2 :Kangchenjunga ... :Shishapangma )
ii) :people rdfs:subclassOf [rdf:type owl:Restriction;
     owl:maxcardinality "1"^^xsd:nonNegativeInteger;
     owl:onProperty :birthdate]
```

The second example can be generalized. Asserting local completeness is often possible through additional properties about the *cardinalities* for SP object sets. By adding statements such as

$\langle \texttt{Bob Dylan}, \texttt{number of albums}, 51\rangle$

it is easy to compare the cardinality of

$\{O \mid \langle \texttt{Bob Dylan}, \texttt{released album}, O\rangle\}$

against the asserted number. A perfect match indicates local completeness. If the object set is smaller, some knowledge is missing; if it is larger than the stated number, some of the statements are wrong (or the statement about `number of albums` is incorrect or stale). A systematic study of such counting statements and their underlying object sets has been performed by [184].

8.2.2 Negative Statements

The design philosphy of KBs is to store positive statements only: facts that do hold. However, it is sometimes also of interest to make negative statements explicit: statements that do not hold, despite common belief or when they are otherwise noteworthy. For example, we may want to explicitly assert that *Ennio Morricone has never lived in the USA*, despite his great success and influence in Hollywood, or that *Elvis Presley has not won an Oscar* despite starring in a number of movies and having been the idol of an entire generation. In addition to such fully grounded statements, the absence of objects for a given property could be of interest, too. For example, *Angela Merkel has no children*. Having this knowledge at hand can help question answering (e.g., to correct high-scoring spurious answers when the proper result is the empty set) as well as KB curation (e.g., to refute improper insertions based on common misbeliefs).

Negative assertions are straightforward to express in logical terms:

$\neg\ livedIn\ (Ennio\ Morricone, USA)$,

$\neg\ wonPrize\ (Elvis\ Presley, Academy\ Award)$,

$\neg\ \exists\ O:\ hasChild\ (Angela\ Merkel, O)$.

The OWL 2 standard [601] provides syntax to capture such formulas. However, specific

KBs like Wikidata have only limited ways of expressing negative statements in a principled and comprehensive manner. Wikidata, for example, has statements of the form

⟨ `Angela Merkel`, `child`, `no value`⟩

where `no value` has the semantics that no object exists (in the real world) for this SP pair. Essentially, this asserts the LCA for this local context, confirming that the absence of objects is indeed the truth (as opposed to merely not knowing any objects). In addition, statements about counts such as

⟨ `Angela Merkel`, `number of children`, 0⟩

can capture empty sets as well. The negation of fully grounded statements is not expressible in RDF, whereas it is straightforward in the OWL 2 language [601].

Obviously, it does not make sense to add *all* (or too many) negative statements even if they are valid. For example, Elvis Presley did not win the Nobel Prize, the Turing Award, the Fields Medal etc. But these statements are not interesting, as nobody would expect him to have these honors. So a key issue is to identify *salient negative statements* that deserve being made explicit.

The work in [16] developed a number of techniques to this end. One of the approaches is to compare an entity against its peers of similar standing, for example, Ennio Morricone against other famous contributors of Hollywood productions. If we see that many of these peers have lived in the USA, then Morricone's counterexample is a remarkable negative statement. Of course, the negative statement could as well be wrong; that is, Morricone has lived in the USA but we have missed out on this fact so far. To rectify this situation, the LCA can be asserted for properties of interest, allowing only the prediction of negative statements if at least one positive statement is present. This high-level idea can be made quantitative based on statistical arguments, leading to a ranked list of salient negative statements for a given entity (see [16] for details).

8.3 Rules and Constraints

So far, we have considered only atomic SPO statements such as `married (ElvisPresley, PriscillaPresley)`. An important asset for curation are logical patterns, or ideally **invariants** in the KB, such as:

Every child has two parents (at the time of birth).
Mothers are (mostly) married to the fathers of their children.
Only humans can marry.

Such invariants are called *intensional knowledge*. There are different ways to make use of intensional knowledge.

Constraints define invariants that must be satisfied by the KB to be *logically consistent* and fulfilling a necessary condition for being *correct*. For example, whoever is married must be a human person. If the KB knows about Elvis's marriage but does not have him in the class `person` yet, we flag the KB as inconsistent. This is the purpose of constraint languages such as SHACL, discussed in Section 8.3.1.

Rules define a calculus to deduce additional statements, to complete the KB and make it logically consistent. For example, if the KB misses out on Elvis being a `person`, a rule could add him to this class upon the premise that there is a marriage statement about him. This is the purpose of logical rule languages such as OWL, discussed in Section 8.3.2. These languages can also be used to enforce constraints, although with some restrictions.

Soft rules express plausibility restrictions, holding for most cases but tolerating exceptions. For example, the pattern that mothers are married to the fathers of their children certainly has exceptions, but would still capture the common case.
Such rules can serve as either *soft constraints*, to detect implausible content in the KB and flag it for curation, or *soft deduction rules*, to infer additional statements that are likely valid and could expand the KB.
Soft rules can be derived from logical patterns in the KB. This is the purpose of **rule mining** algorithms, discussed in Section 8.3.3.

Hard and soft constraints are a powerful instrument for KB cleaning: identifying implausible and dubious statements, and resolving conflicts and gaps in the KB to establish logical consistency. Methods for this purpose are discussed in Section 8.5.

8.3.1 Constraints

Consistency constraints *prescribe* invariants in the KB: people must have a birth date, people cannot have more than one birth date, etc.

There is a wide spectrum of invariants that we could consider. The following lists some useful templates, by examples.

- **Type constraints**, e.g.:
  $\forall x, y : composed(x, y) \Rightarrow type(x, musician)$
  (domain of the `composed` relation) and
  $\forall x, y : composed(x, y) \Rightarrow type(y, song)$
  (range of the `composed` relation)

- **Value constraints**, e.g.:
  $\forall x, v : type(x, basketballer) \wedge height(x, v) \Rightarrow v < 250cm$
- **Relation restrictions** like symmetry or transitivity, e.g.:
  $\forall x, y : spouse(x, y) \Rightarrow spouse(y, x)$
  (symmetry of the `spouse` relation)
- **Functional dependencies**, e.g.,:
  $\forall x, y, z : birthplace(x, y) \wedge birthplace(x, z) \Rightarrow y = z$
  (the `birthplace` relation is a function)
- **Conditional functional dependencies**, e.g.:
  $\forall x, y, z : citizen(x, y) \wedge citizen(x, z) \wedge y = Germany \Rightarrow y = z$
  (Germany does not allow dual citizenships)
- **Inclusion dependencies**, e.g.:
  $\forall x, y : type(x, singer) \Rightarrow type(x, musician)$

More advanced kinds of constraints include:

- **Disjointness constraints**, e.g.:
  $\forall x, y : type(x, weightlifter) \Rightarrow \neg type(x, ballerina)$ and
  $\forall x, y : type(x, ballerina) \Rightarrow \neg type(x, weightlifter)$
- **Existential dependencies**, e.g.:
  $\forall x : type(x, scholar) \Rightarrow$
  $\exists y : (published(x, y) \wedge type(y, scientific\ article))$
- **Temporal dependencies**, e.g.:
  $\forall x, y, z, s, t : marriage(x, y, s) \wedge marriage(x, z, t) \Rightarrow$
  $(y = z \vee \neg overlaps(s, t))$
  where `marriage` is a ternary relation and s and t are time intervals during which the marriages last.

Some of these invariants may appear too strict, given that real life holds many surprises. For example, there could be exceptions for the mutual exclusion of being a weightlifter and being a ballerina. It is up to the scope and purpose of the KB whether we want to rule out such exceptions or not. Constraints that cover the prevalent case but do tolerate exceptions are very useful; they can be used to flag suspicious statements in a KB and prompt a human curator (see Section 8.5).

Database languages like SQL support the declarative specification of constraints (see textbooks such as [588, 432]). However, knowledge bases have mostly adopted Web standards. The W3C standard language for expressing constraints is the **Shapes Constraint Language (SHACL)** [281]. Here is an example:

```
:Person
  rdf:type owl:Class, sh:NodeShape ;
  :property [
    sh:path :hasMother ;
    sh:maxCount 1 ;
    sh:Class :Person ;
  ] .
```

This code specifies that every instance of the type `Person` has at most one entity in the range of the `hasMother` relation, disallowing more than one biological mother. To enforce exactly one mother in the KB, an analogous shape constraint with `minCount` could be added. However, a SHACL validation tool would then raise an inconsistency flag if some people miss the respective statements for `hasMother`. Therefore, to allow for leeway in the KB growth process, with people being added without complete properties, the `maxCount` constraint only would be a typical design choice.

More advanced features of SHACL allow specifying constraints on strings (such as length restrictions and compliance with regex patterns) for attributes, value ranges for numerical properties, and composite constraints via Boolean operators [281]. An alternative to SHACL is the **Shape Expressions Language (ShEx)** [182]. ShEx comes with a validation algorithm to test whether all constraints hold in a given KB [52].

8.3.2 Deductive Rules

A deductive rule can infer statements for addition to a KB, to make it consistent and to enhance its coverage. For example, a rule can codify that the entities in a `marriedTo` relationship must be human people; so `Elvis Presley` would be added to the class `person`. Rules may produce contradictions among their deduced statements, though. For example, an additional rule could state that entities must not belong to both the class of persons and the class of fictitious characters. If the KB already identifies Elvis as an instance of `fictious character` then would derive a contradiction. However, unlike constraints discussed in Section 8.3.1, this would not automatically result in the explicit removal of contradictory statements.

Deduction rules are expressed in appropriately chosen fragments of first-order predicate logics, typically trading off expressiveness versus computational complexity. In the Semantic Web world, several logics are widely used. **RDFS (= RDF Schema)** is the simplest formalism [604]. It allows only limited kinds of rules: domain and range rules, subclass-of rules, and sub-property rules. In RDFS, we can specify that everyone who is married is a person:

```
:marriedTo rdfs:domain :Person
```

RDFS cannot express the disjointness of classes. For this, a more powerful language has been developed: the **Web Ontology Language OWL 2** [601], which allows asserting that certain classes rule each other out. OWL exists in several flavors, from the least expressive (and more computationally benign) to the most expressive (and computationally expensive) variant. For our purpose, the OWL 2 QL flavor is sufficient to specify that real-life persons and fictitious characters are disjoint:

```
:Person owl:disjointWith :FictitiousCharacter
```

OWL reasoners can ingest sets of formulas to perform deduction, and flag the KB as inconsistent if a contradiction is found. The theoretical foundation of OWL is **description logics** [551, 22]. Our example can be expressed as follows:

$FictitiousCharacter\ (Elvis)$
$marriedTo\ (Elvis, Priscilla)$
$\exists\ marriedTo \sqsubseteq Person$
$Person \sqcap FictitiousCharacter \sqsubseteq \bot$

The OWL reasoner would detect that this has a logical contradiction. There are many approaches that aim to repair such contradictions [37, 38]. One option is to identify a minimal set of axioms to remove. Alternatively, we can compute the maximal subset of instance-level statements that are still compatible with all the given axioms. In the example, we can remove either the first statement or the second. There are different ways to prioritize the statements, for example, by preferably keeping those with high confidence [44].

8.3.3 Rule Mining

Rule mining is the task of identifying logical patterns and invariants in a knowledge base. For example, we aim to find that people who are married usually live in the same city:

$\forall x, y, z :\ marriedTo(x, y) \wedge livesIn(x, z) \Rightarrow livesIn(y, z)$

or that mayors of cities have the citizenship of the respective countries:

$\forall x, y, z :\ mayor(x, y) \wedge locatedInCountry(y, z) \Rightarrow citizen(x, z)$

Rule mining can be used either for learning *constraints* to prune out spurious statements, or for *rule-based deduction* to fill gaps in the KB. Typically, the same rule is used for either one of two purposes; this is the choice of the KB architect, depending on where the pain points are: precision or recall.

Rules are often of soft nature, holding for a large fraction of statements and tolerating "minorities". Thus, rule mining needs to consider metrics for the degree of validity: *support* and *confidence*, as discussed below. In addition to playing a vital part in the KB curation process, rules can also give insight into potential biases in the KB content and, possibly, even the real world. For example, a rule saying that *actors are (usually) millionaires*

may arise from the KB emphasis on successful stars and lack of covering long-tail actors, and a rule saying that

Nobel laureates are (mostly) men

reflects the gender imbalance in our society. Yet another potential use case for rule mining is to generate *explanations* for doubtful facts – in combination with retrieving evidence or counter-evidence from text corpora (see, e.g., [173]).

Rule mining, as discussed in the following, is related to the task of discovering *approximate functional dependencies* and *approximate inclusion dependencies* in relational databases (e.g., [257, 88, 514]), as part of the data cleaning pipeline or for data exploration. The textbook [255] covers this topic, including references to state-of-the-art algorithms. Note that functional and inclusion dependencies are special cases of logical invariants. KB rule mining also aims at a broader, more expressive class of logical patterns, such as Horn rules.

Horn Rules and Inductive Logic Programming:
Unless we restrict rules to suitable fragments of first-order logics, rule mining is bound to run into computational complexity issues and intractability. Therefore, the rules under consideration are usually restricted to the following shape:

A **Horn rule** is a first-order predicate logic formula restricted to the format

$\forall x_1 \forall x_2 \ldots \forall x_k :$

$P_1(args_1) \wedge P_2(args_2) \wedge \cdots \wedge P_m(args_m) \Rightarrow Q(args)$

where x_1 through x_k are variables, P_1 through P_m and Q are KB properties, and their arguments $args1$ through $args_m$ and $args$ are ordered lists of either variables from $\{x_1 \ldots x_k\}$ or constants, that is, entities or literal values from the KB. The properties are often binary (if the KB adopts RDF) but could also have higher arities.

The conjunction of the terms $P_1(args_1)$ through $P_m(args_m)$ is called the **rule body**, and $Q(args)$ is called the **rule head**. The terms $P_i(args_i)$ and $Q(args)$ are called **atoms** (or *literals*).

As all quantifiers are universal (i.e., no existential quantifiers) and prefix the propositional-logic part of the formula (i.e., using prenex normal form), we often drop the quantifiers and write

$P_1(args_1) \wedge P_2(args_2) \cdots \wedge P_m(args_m) \Rightarrow Q(args)$.

When normalized into **clause** form with disjunctions only, the propositional-logics part looks like

$\neg P_1(args_1) \vee \neg P_2(args_2) \cdots \vee \neg P_m(args_m) \vee Q(args)$.

Horn clauses restrict formulas to have at most one positive atom. The two examples at the begin of this subsection are Horn rules.

Automatically computing such rules from extensional data is highly related to the task of **Association Rule Mining** [5, 4, 207], for example, finding rules such as:

Customers who buy white rum and mint leaves
will also buy lime juice (to prepare Mojito cocktails).

or

Subscribers who like Elvis Presley and Bob Dylan
will also like Nina Simone.

However, association rules, mined over transactions and other user events, do not have the full logical expressiveness. Essentially, they are restricted to single-variable patterns, capturing customers, subscribers, users etc., but no other variables. The logics-based generalization of rules has been addressed in a separate community, known as **Inductive Logic Programming** [407, 654, 102]. Both association rule mining and inductive logic programming face a huge combinatorial space of possible rules. To identify the interesting ones, quantitative measures from data mining need to be considered:

> Given a rule $B \Rightarrow H$ over a knowledge base KB and a substitution θ that instantiates the rule's variables with entities and values from KB, the resulting instantiation of the rule head, $\theta(H)$, is called a **prediction** of the rule. A prediction is called *positive* if the grounded head is contained in the KB, and *negative* otherwise. We refer to these also as *positive and negative samples*, respectively.
> The **support** of a Horn rule is the number of positive predictions: $support(P_1 \ldots P_m \Rightarrow Q(args)) =$
> $$|\{\theta(args) | P_1 \ldots P_m, Q \text{ and } Q(\theta(args)) \in KB\}|$$
> The **confidence** of a Horn rule is the fraction of positive predictions relative to the sum of positive and negative predictions:
> $$confidence(P_1 \ldots P_m \Rightarrow Q(args)) = \ support \ / \ (support \ + \ |\{\theta(args) | P_1 \ldots P_m, Q \text{ and } Q(\theta(args)) \notin KB\}|)$$

Rule Mining Algorithms:
The goal of KB rule mining is to compute high-confidence rules, but to ensure significance, these should also have support above a specified threshold. Finding rules can proceed in two ways: bottom-up or top-down.

Top-down algorithms start with the head of a rule and aim to construct the rule body by incrementally adding and refining atoms – so that rules gradually become more specific. *Bottom-up algorithms* start from the data, the KB content, and select atoms for the body of initially special rules that are gradually generalized. Both of these paradigms resemble key elements of the A-Priori algorithm for association rules [5], namely, iteratively growing the atoms sets for rule bodies (like in itemset mining), while checking for sufficiently high support and scoring the rule confidence. In particular, all algorithms exploit the anti-monotonicity of the support measure, so as to prune candidate rules that cannot exceed the support threshold.

The following algorithmic skeleton outlines the **AMIE** method [174, 177, 305] for top-down rule mining with binary predicates (i.e., over SPO triples), using a queue for atom sets that can form rule bodies.

Top-down mining of KB rules:
Input : head predicate Q(x,y) with variables x,y, and the KB
Output : rules with support and confidence

0. Initialize a queue with empty set
1. Pick an entry $\mathbf{P} = \{P_1 \dots P_l\}$ from the queue and extend it by generating an additional atom P_{l+1} where
   a. P_{l+1} shares one argument with those of $\mathbf{P}$ or Q and has a new variable for the other argument, or
   b. P_{l+1} has a variable as one argument shared with those of $\mathbf{P}$ or Q and introduces a new constant (entity or value) for the other argument, or
   c. P_{l+1} shares both of its arguments with those of $\mathbf{P}$ or Q
2. Compute confidence, support and other measures for the rule $P_1 \wedge \dots \wedge P_{l+1} \Rightarrow Q$
3. Insert atom set $\{P_1 \dots P_l, P_{l+1}\}$ into the queue
   if support above threshold
4. Output the rule $P_1 \wedge \dots \wedge P_{l+1} \Rightarrow Q$
5. Repeat steps 1–4
   until enough rules or priority queue exhausted

Instead of a simple queue, we can also use a priority queue by confidence score, a combination of confidence and support, or other measures of interestingness, so as to arrive at the most insightful rules for the final output. Recall that confidence is the ratio of correct predictions and the sum of correct and incorrect predictions, where a prediction is an instantiation of the rule head in line with the variable bindings in the rule body. Confidence can be calculated in different ways in a KB setting. The difference lies in selecting negative cases (i.e., what counts as an incorrect prediction).

1. By postulating the **Closed-World Assumption (CWA)**, we identify instantiations of the rule body such that the accordingly instantiated head is not in the KB. Figure 8.1 shows an example for deducing fathers of children. All predicted fathers for which no statement is in the KB count as negative samples. This way, the number of negative samples is typically high, as most KBs are far from being complete. So the confidence tends to be inherently underestimated.
2. To counter the biased influence of absent statements being counted as negative samples, we can alternatively adopt the **Local Completeness Assumption (LCA)**, as defined

in Section 8.2. Missing statements are considered as negative evidence only if they have at least one statement for the property in the respective rule atom (e.g., `fatherOf`). Otherwise, the absent statements are disregarded, as their validity is uncertain under the default Open World Assumption. Figure 8.1 shows an example.

∀x,y,z: married(x,y) ∧ motherOf(x,z) ⇒ fatherOf(y,z)		
Married(Priscilla,Elvis)	married(Michelle,Barack)	married(Diana,Charles)
motherOf(Priscilla,Lisa)	motherOf(Michelle,Malia)	motherOf(Diana,William)
	motherOf(Michelle,Sasha)	motherOf(Diana,Harry)
fatherOf(Elvis,Lisa)		fatherOf(Charles,William)
CWA positive predictions:		
fatherOf(Elvis,Lisa)		fatherOf(Charles,William)
CWA negative predictions:		
	fatherOf(Barack,Malia)	fatherOf(Charles,Harry)
	fatherOf(Barack,Sasha)	
	CWA-based confidence: 2/5	
LCA positive predictions:		
fatherOf(Elvis,Lisa)		fatherOf(Charles,William)
LCA negative predictions:		
		fatherOf(Charles,Harry)
	LCA-based confidence: 2/3	

Figure 8.1: Example for rule confidence under CWA and LCA

As shown by [174, 177], the LCA-based confidence measure is more effective in identifying interesting rules by confidence scoring. Note that the mined rules may include constants for some arguments, for salient values or entities. Examples are:

$\forall x, y : type(x, musician) \ \wedge \ livesIn(x, Nashville) \Rightarrow citizenOf(x, USA)$
$\forall x, y : type(x, politician) \ \wedge \ citizenOf(x, France) \Rightarrow livesIn(x, Paris)$

By swapping the positive examples and the negative examples, it is also possible to learn rules with negation. For example, if two people are married, then one cannot be the child of the other. An issue here is that if LCA were used to generate negative samples, it would yield an infinite number of negative statements: Elvis is not the child of Madonna, Elvis is not the child of Barack Obama, etc. To consider only a finite number of meaningful cases, the solution is to generate a negative sample $Q(x, y)$ only if x and y appear together in at least one SPO statement in the KB [434, 433]. More precisely, we generate the statement $Q(x, y)$ as an incorrect prediction if

1. $Q(x, y)$ is not in the KB.
2. There is y' with $Q(x, y')$ in the KB or there is x' with $Q(x', y)$.

3. There is a predicate R with $R(x, y)$ in the KB.

A variety of works have developed algorithmic alternatives, extensions and optimizations, most importantly, for pruning the search space and for avoiding expensive computations of support and confidence to the best possible extent. Relevant literature includes [310, 308, 579, 376, 305]. A particularly noteworthy method is the **Path Ranking Algorithm** by [310, 308] that operates on a graph view of the KB, with entities as nodes and properties as edges. The algorithm computes frequent edge-label sequences on the paths of this graph; these form candidates for the atom sets of rules. Random walk techniques are leveraged for efficiency.

Several works have investigated generalizations beyond Horn rules: mining rules with exceptions [172], OWL constraints [599], rules with numerical constraints [388], rules in combination with text-based evidence and counter-evidence [173], and rules that take into account the confidence scores of other rules [376]. Surveys on KB rule mining are given by [554] and [564].

8.4 Knowledge Graph Embeddings

So far, we have looked at symbolic representations of entities, types and properties, founded on computational logics. Recently, another kind of knowledge representation has gained popularity: mapping entities, properties and SPO triples into real-valued vectors in a low-dimensional latent space. Such **knowledge graph embeddings**, or **KG embeddings** for short, are motivated by the great success of word embeddings in machine learning for NLP [384]. Note that the setting here is different from those for word embeddings and entity embeddings such as Wikipedia2vec [638], as covered in Section 4.5. The focus here is on capturing structural patterns of an entity in its graph neighborhood alone, as an asset for machine learning and predictions.

Imagine a KB where a set of entities of type `person` and the relation `spouse` have all been mapped to latent vectors, as illustrated in Figure 8.2 with the simplified case of a two-dimensional latent space. The dots represent the coordinates of these vectors. The `spouse` vector has been shifted and scaled in length, while retaining its direction, to connect entities which are known to be spouses. Now consider the French president `Emmanuel Macron` and a set of potential wives such as `Carla Bruni`, `Francoise Hardy` and `Brigitte Macron`. We can identify `Brigitte Macron` as his proper spouse by adding up the vectors for `Emmanuel Macron` and the `spouse` relation, as illustrated in the figure.

Use Cases of KG Embeddings:
The vector representation has several use cases. It is a good way of quantifying similarities between entities, and thus enables statistical methods like clustering. Also, deep learning requires real-valued vectors as input. The most prominent application is to **predict miss-**

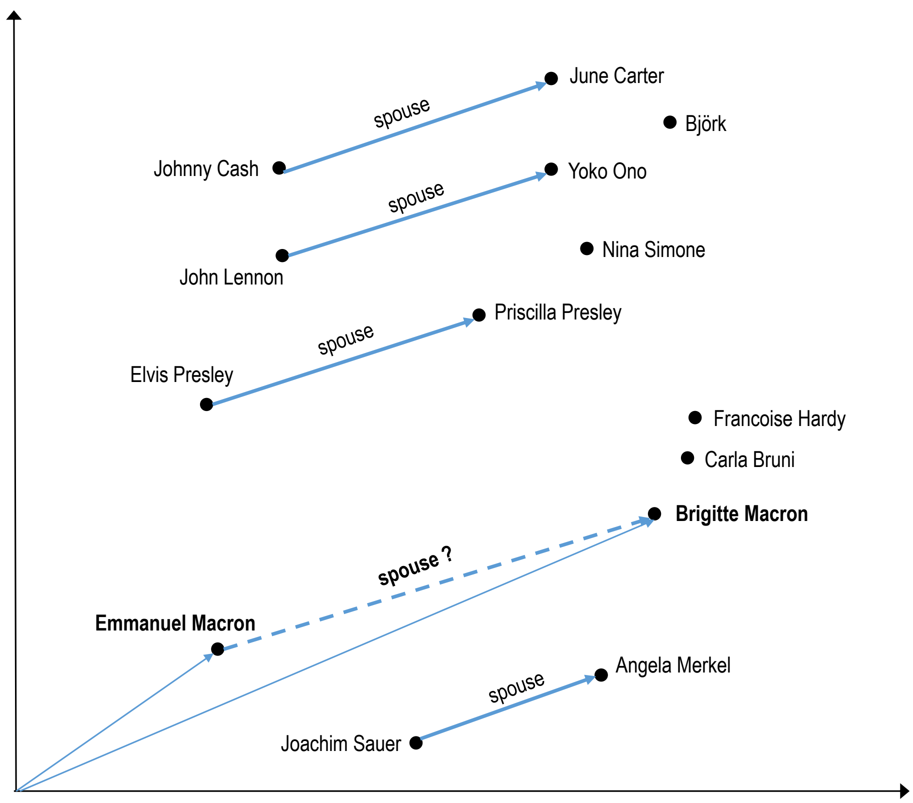


Figure 8.2: Embeddings of entities and the spouse relation

ing statements for the KB. The literature has referred to this as **knowledge graph completion** [424, 54, 543, 341]. By operating on structural patterns of the graph alone, this is actually a form of *link prediction*, similar to predicting (and recommending) "friends" in a social network. Specifically, for a given target entity and relation, KG completion would compute a ranked list of candidates for the respective object(s) to suggest a new SPO triple. This could support a KB curator in filling gaps. However, state-of-the-art methods for KG embeddings are still far from consistently recommending the correct entities at the top ranks. So human curators do not run out of work that soon.

A dual case is to scrutinize doubtful statements in the KB, or in a pool of candidates considered for addition. If none of the say top-100 predictions for a given SP pair produces the O that appears in a candidate statement, we should consider discarding the candidate completely. However, this should still be taken with a big grain of salt when maintaining a production-level KB.

Learning of KG Embeddings:
The objective for learning embedding vectors is as follows:

KG Embedding Vectors:
Given a KB, find a mapping v from its entities and relations onto real-valued vectors, such that

$$v(s) + v(p) \approx v(o)$$

for each statement $\langle s, p, o \rangle$ in the KB.
The application then is, for given s and p with unknown o, to predict o by finding objects close to $v(s) + v(p)$.

Embeddings can be computed by algorithms for tensor factorization (e.g., [424]) or by training a neural network (e.g., [54, 543]). At training time, the neural network takes as input $\langle s, p, o \rangle$ statements that are one-hot encoded as preliminary vectors. The network learns latent vectors $v(s), v(p), v(o)$ and outputs a score how well these learned embeddings approximate the existing facts. So the key term in the loss function is $||v(s) + v(p) - v(o)||$ summed up over all training samples. To minimize the loss function, the learner performs backpropagation with gradient descent, and this way computes the best embedding vectors. At deployment time, one can simply add or subtract learned vectors to compute similarities between vector expressions.

To avoid that the network just returns 0, that is, perfect matches, for all inputs, it has to include also *negative samples* for its training. These are usually obtained by perturbing positive statements $\langle s, p, o \rangle$ of the KB into statements $\langle s, p, o' \rangle$ that are not in the KB. For negative training samples, the network should return a large output value (i.e., mismatch).

This basic method is known by the name **TransE** [54], for "translating" entities into latent-space vectors. A major limitation of TransE is that all objects of a one-to-many relation end up in the same spot in the vector space. In our example, if Emmanuel Macron ever marries someone else and the marriage triples are the only statements about his spouses, then that other person would have the same vector as Brigitte Macron (much to her annoyance, certainly). To overcome this and other limitations, more advanced models have been developed. Major works, overviews and experimental comparisons include [424, 543, 54, 341, 423, 611, 266, 503].

8.5 Consistency Reasoning for Knowledge Cleaning

Constraints allow us to scrutinize each statement candidate, one at a time. For example, a candidate statement that Donald Trump has composed the US national anthem "Star Spangled Banner" could be refuted by violating the type constraint that pieces of music must be created by musicians, or at least artists. For doing this, we only need to check this *individual candidate* against the applicable constraints, without looking at any of the other candidates.

However, the situation is often more complex. Suppose, for example, that we have three candidates for the place where Elvis Presley died:

a) ⟨`Elvis Presley, deathplace, Baptist Hospital Memphis`⟩,
b) ⟨`Elvis Presley, deathplace, Graceland Ranch`⟩,
c) ⟨`Elvis Presley, deathplace, Mare Elysium (Mars)`⟩
(where he lived until February 2020, according to some beliefs).

Each of them individually could be accepted, but together they violate the constraint that the deathplace of a person is unique. So we must inspect the evidence and counter-evidence for the three statements *jointly*, to arrive at a conclusion on which of these hypotheses is most likely to be valid. The following subsections present various methologies for the necessary *holistic* consistency checking.

8.5.1 Data Cleaning and Knowledge Fusion

Data Cleaning with Minimal Repair:
Although relational databases usually undergo a careful process for schema design and data ingest, the case where some data records contain erroneous values is a frequent concern. Humans may enter incorrect values, and, most critically, database tables may be the result of some imperfect data integration, for example, merging two or more customer tables with discrepancies in their addresses. A major part of the necessary *data cleaning* [472, 256, 255], therefore, is *entity matching (EM)* [287, 416, 94, 135, 96]: inferring which records truly correspond to the same entities and dropping or correcting doubtful matches. We discussed this issue in Section 5.2.

Entity-matching errors are not the only issue in data cleaning, though. A typical setting involves a relational table and a set of integrity constraints that the data should satisfy. These constraints could be manually specified, but could also be automatically discovered from the data itself – by data-mining algorithms that reveal *approximate invariants* [257, 88, 514]. These include functional dependencies (FDs), inclusion dependencies (IDs), conditional FDs, and more. The family of *denial constraints* subsumes a variety of these kinds of invariants, and has been used in industrial-strength data-cleaning tools [486, 255]. The key idea for cleaning is: when a constraint holds for almost all of the data instances, the remaining violations are likely errors. To remove these errors, we either need a way of obtaining the correct values, possibly by some human-in-the-loop procedure, or we remove the incorrect records. Often, there are multiple ways of choosing a set of erroneous records for removal, in order to restore the integrity of the remaining data. The guiding principle in making good choices is the following [14, 49, 38]:

Minimal-Repair Data Cleaning:

- Input: Relational table R with errors, and a set of constraints
- Output: Clean table $S \subset R$
  Choose tuples $\{t_1, t_2 \ldots\}$ such that
  $S = R - \{t_1, t_2 \ldots\}$ satisfies all constraints, and
  $|\{t_1, t_2 \ldots\}|$ is mimimal.

The repair mechanism can vary: instead of solely considering discarding entire tuples, there are alternative options, including the replacement of erroneous attribute values. The literature on database (DB) cleaning has developed a suite of powerful algorithms for minimum repair [255]. They include both combinatorial optimization methods, with smart pruning of the underlying search space (e.g., [99, 462]), and machine-learning methods, such as probabilistic graphical models (e.g., [486]). We will come back to the latter in Section 8.5.4.

In principle, DB cleaning methods are applicable to knowledge bases as well, and could contribute to KB curation. However, this has not been explored much, for several reasons. First, data cleaning typically tackles a single relational table, whereas KBs cover hundreds or thousands of relations. Second, at this magnitude of the schema size, the number of constraints can be huge and DB methods are not geared for this scale (at the schema level). Third, the assumption for DB cleaning is that most records are correct and errors are spread among a few incorrect values of single attributes. In contrast, KB curation may start with a huge number of uncertain statements, all collected by potentially noisy extraction algorithms. It is well possible that a KB-cleaning process would have to remove ten, twenty or more percent of its inputs.

Multi-Source Fusion:

Another related topic in database research is **data fusion** [134, 128]: given a set of value-conflicting tuples from different databases or structured web sources, the task is to infer the correct value. For illustration, consider the data about people's current residence, with values obtained from four different sources.

	Source A	Source B	Source C	Source D
Fabian	Paris	Paris	Mountain View	Paris
Gerhard	Seattle	Saarbruecken	Kununurra	Saarbruecken
Luna	Mountain View	Seattle	Mountain View	Seattle
Simon	Saarbruecken	Bolzano	Saarbruecken	Bolzano

The goal is to choose the correct residence values for each of the four people. A simple approach could aim to identify the *best source*, by some authority or trust measure (e.g., Alexa rank for web sites), and pick all values from this source. In the example, Source B could be the best choice. However, this alone would yield an incorrect value for Simon. Therefore, an alternative is to compute a *voting* from all sources, and let the majority win. However, this would yield ties for Luna and Simon. So the best approach is to combine *source quality* and *weighted voting*, with source quality cast into weights.

The key point now is how to measure source quality. Suppose, for ease of explanation, that we have ground-truth values for a subset of the tuples, for example, by considering some of the sources as a priori trustworthy, and applying the fusion algorithm to all other sources and their additional data values. Then, the source quality can be defined as the fraction of its values that match the ground-truth. In the example, if we knew all correct values, source A would have weight 2/4, source B 3/4, source C 1/4 and source D 3/4.

The solution devised in [128] is based on this intuition, but does not require any ground-truth knowledge. The approach uses voting to estimate a prior for trustworthy sources, and then applies Bayesian analysis with all observed data to infer the most likely valid values. This method has been generalized and carried over to **knowledge fusion** [130, 131]:

Multi-Source Fusion for KB Statements:
Input: set of uncertain SPO statements $\{t_1, t_2 \dots t_n\}$
from various sources $\{S_1 \dots S_k\}$ with $k \ll n$,
where each S_j stands for a combination of website and extraction method (e.g., regex-based extraction from lists in musicbrainz.org)
Output: jointly learned probabilities (or scores) for

- the correctness of statement t_i $(i = 1..n)$ and
- the trustworthiness of source S_j $(j = 1..k)$.

The joint learning of statement validity and source trustworthiness takes into account *mutual reinforcement dependencies*. A specific instantiation, developed in [131], is based on a sophisticated equation system and an iterative EM algorithm (EM = Expectation Maximization) for an approximate solution.

Methods along these lines have been utilized in the Google Knowledge Vault project [129]. Section 9.5.2.2 provides further insights on this work.

8.5.2 Constraint Reasoning and MaxSat

Knowledge fusion and other techniques for ensemble learning still treat each candidate statement in isolation. For example, they make independent decisions about the truth of ⟨`Bob Dylan, citizenship, USA`⟩ and ⟨`Bob Dylan, citizenship, UK`⟩. However, it is often beneficial to perform *joint inference* over a set of candidates, based on *coupling constraints*. In the following, we present a major line of how to do this, using so-called *Weighted MaxSat* inference, which is an extension of the *Maximum Satisfiability* problem, MaxSat for short [45]. A broader perspective would be to harness methods from *constraint programming* [496], but we focus on MaxSat here as it has a convenient way of incorporating the *uncertainty* of the inputs.

Consider, for example, a set of candidate statements *hasWon(BobDylan,Grammy), hasWon(BobDylan,LiteratureNobelPrize)*, and (by erroneously extracting the type of Nobel Prize) *hasWon(BobDylan,PhysicsNobelPrize)*, along with two type statements *type(BobDylan,musician)* and *type(BobDylan,scientist)*. Imposing consistency constraints is a good way of corroborating such uncertain candidates. For the simplified example, we postulate that people are either scientists or musicians (i.e., never both), and that it is unlikely for a musician to win a Nobel Prize in Physics. These (soft) constraints can be either specified by a knowledge engineer or automatically learned (e.g., by rule mining, see Section 8.3.3). The following table shows the five noisy candidate statements and the constraints (with numbers in brackets denoting weights, explained below).

⟨`BobDylan hasWon Grammy`⟩ [0.7]
⟨`BobDylan hasWon LiteratureNobelPrize`⟩ [0.5]
⟨`BobDylan hasWon PhysicsNobelPrize`⟩ [0.3]
⟨`BobDylan type musician`⟩ [0.9]
⟨`BobDylan type scientist`⟩ [0.1]
$\forall x (hasWon(x, Grammy)) \Rightarrow type(x, musician))$ [0.9]
$\forall x (hasWon(x, PhysicsNobelPrize)) \Rightarrow type(x, scientist))$ [0.9]
$\forall x (type(x, scientist)) \Rightarrow \neg type(x, musician))$ [0.8]
$\forall x (type(x, musician)) \Rightarrow \neg type(x, scientist))$ [0.8]

All of these logical formulas together are unsatisfiable; that is, they are inconsistent and imply *false*. However, if we choose a proper subset of them, we may arrive at a perfectly consistent solution and would thus identify the statements that we should accept for the KB. Typically, this would only involve discarding some of the atomic statements while retaining all constraints. However, it is also conceivable to drop constraints, as they are soft and could be violated by exceptions. Obviously, we want to sacrifice as few of the inputs as possible, maximizing the remaining proper statements. This problem of computing a large consistent subset of the input formulas is the **Maximum Satisfiability** problem, **MaxSat** for short.

Both candidate statements and constraints often come with weights; these are shown in brackets in the above table. For statements, the weights are usually the confidence scores returned by the extractor, potentially with re-scaling and normalization. For constraints, the weights would reflect confidence scores for underlying logical invariants (see Section 8.3.3), or the degrees to which they should be fulfilled in a proper KB. Another way of interpreting weights is that they denote *costs* that a reasoner has to pay when dropping a statement or violating a constraint. The goal is now refined as follows:

> **Principle of Weighted Maximum Satisfiability**:
> For a given set of weighted candidate statements and weighted consistency constraints, identify a subset of formulas that is logically consistent and has the highest total weight.

Intuitively, in the example, we can discard the statement that Dylan won the Nobel Prize in Physics and the statement that he is a scientist, which come at a cost of 0.3 and 0.1, respectively. The remaining formulas are then logically consistent.

Grounding into Clauses:
Our input so far is heterogenous, mixing apples and oranges: candidate statements are propositional-logic formulas without variables, whereas constraints are predicate-logic formulas with variables (and quantifiers like $\forall$ and perhaps also $\exists$). To cast the consistency reasoning into a tangible algorithmic problem, we need to unify these two constituents. First, we limit constraints to be of the *Horn clause* type, in prenex normal form with universal quantifiers only (see Section 8.3.3). By being more permissive on the number of positive atoms, we can further relax this into arbitrary *clauses*.

Second and most importantly, we *instantiate* the constraints by substituting variables with constants from the statements. For example, we plug in `BobDylan` for the x variable in all constraints. If we had more candidate statements, say also about `ElvisPresley` and `EnnioMorricone`, we would generate more instantiations. This renders all formulas into propositional logic without any variables, and all formulas become clauses. In computational logics, this process is referred to as **grounding**. For our simple example, the grounding would produce the following set of clauses:

⟨`BobDylan hasWon Grammy`⟩ [0.7]
⟨`BobDylan hasWon LiteratureNobelPrize`⟩ [0.5]
⟨`BobDylan hasWon PhysicsNobelPrize`⟩ [0.3]
⟨`BobDylan type musician`⟩ [0.9]
⟨`BobDylan type scientist`⟩ [0.1]
¬⟨`BobDylan hasWon Grammy`⟩ ∨ ⟨`BobDylan type musician`⟩ [0.9]
¬⟨`BobDylan hasWon PhysicsNobelPrize`⟩ ∨ ⟨`BobDylan type scientist`⟩ [0.9]
¬⟨`BobDylan type scientist`⟩ ∨ ¬⟨`BobDylan type musician`⟩ [0.8]

The mutual exclusion between types `scientist` and `musician` was written in the form of two implication constraints before, but both result in the same clause which is thus stated only once.

Obviously, a less simplified case could have constraints with multiple variables – so this process has a potential for "combinatorial explosion". The full grounding happens only conceptually, though, and does not have to be fully materialized. Also, if entities have been canonicalized upfront and are associated with types in the KB, then only those combinations for variable substitutions need to be considered that match the type signatures of the constraint predicates. There are further optimizations for lazy computation of groundings.

This way, the consistency reasoning task has been translated into propositional logics, with weighted clauses. We now treat the atoms of all clauses together as statements for which we want to infer truth values. For clauses with more than one atom (i.e., the ones for instantiated constraints), the entire clause becomes satisfied if at least one of its atoms has a positive truth value (with consideration of whether an atom itself is positive or negative, i.e., prefixed by ¬).

> **Objective of Weighted MaxSat**:
> Given a set of propositional-logic clauses, each with a positive weight, the objective of Weighted MaxSat is to compute a truth-value assignment for the underlying atoms such that the total weight of the satisfied clauses is maximal.

In the example, with 8 clauses and a total of 5 atoms, the optimal solution is to assign *false* to the atoms ⟨`BobDylan hasWon PhysicsNobelPrize`⟩ and ⟨`BobDylan type scientist`⟩, with a total weight of 4.7. A different truth-value assignment, which is sub-optimal but consistent, would be to assign *false* to ⟨`BobDylan type musician`⟩ and ⟨`BobDylan hasWon Grammy`⟩, and *true* to the other three atoms. This would have a total weight of 3.5.

Weighted MaxSat Solvers:
Relating this approach to MaxSat reasoning for tasks in computational logics, we treat the atoms of our clauses as variables for truth-value assignment; this way, we can directly use state-of-the-art solvers for weighted MaxSat. The problem is NP-hard, though, as it generalizes the classical SAT problem. Nevertheless, there are numerous approximation algorithms with very good approximation ratios (see, e.g., [326]). Many of these have been designed for use cases like theorem proving and reasoning about program correctness. In these settings, the structure of the input sets is rather different from the KB environment. Theorem proving often deals with a moderate number of clauses where each clause could have a large number of atoms. These are automatically derived from declarative specifications in more expressive logics. For reasoning over KB candidates, where all clauses are either single-atom or generated by grounded constraints, the situation is very different. The grounding can lead to a huge number of clauses, but the atom set per clause is fairly small as the formulas mostly express type constraints, functional dependencies, mutual exclusion, and other compact design patterns.

This design consideration has motivated the development of *customized MaxSat solvers* for KB cleaning. Two simple but powerful heuristics have been used in [566], originally developed to augment the YAGO knowledge base.

The first technique leverages *Unit Clauses*, that is, clauses that have exactly one variable whose truth value has not yet been determined. We compute, for each unassigned variable x, the difference between the combined weight of the unit clauses where x appears positive and the combined weight of the unit clauses where x appears negative. We choose the variable x with the highest absolute difference, and set it to true if the difference is positive and to false otherwise. This technique is a greedy heuristic, which assumes that the variable with the highest value brings a high gain for the final solution.

The method can be combined with another technique known as *Dominant Unit Clause (DUC) Propagation*. DUC propagation fixes the values of variables that are already so constrained that only one truth value can be part of the optimal solution. Specifically, it sets the value of variables that appear with one polarity in unit clauses that have a higher combined weight than all clauses (regardless of unit clause or not) where the variable appears with the opposite polarity. In combination, a SAT solver with these two heuristics has an approximation guarantee of 1/2: solutions have a weight that is at least 50% of the optimal solution. In practice, the approximation is much better, often reaching 90% or better [566]. Also, the algorithm is very efficient and can run on large sets of input clauses – automatically generated from uncertain candidate statements and consistency constraints. For huge inputs, it is even feasible to *scale out* the computation, by partitioning the input via graph-cut algorithms and running the reasoner on all partitions in parallel [412].

Extensions:
The Weighted MaxSat method has been used in the **SOFIE** system [566] to extract information from noisy text. SOFIE reformulates the statement-pattern duality of Section 4.3.2 as a soft constraint between patterns and relations: if a pattern expresses a relation, then two co-occurring entities with this pattern are an instance pair of the relation. Conversely, every sentence that contains two entities that are known to be connected by the relation becomes a candidate for a pattern. In addition, SOFIE has integrated *entity canonicalization* (see Chapter 5) into its joint reasoning, by additional constraints that couple surface names and entities.

The HighLife project [149] has extended MaxSat reasoning to higher-arity relations. Texts can express a relationship that holds between more than two entities. If only some of these entities are mentioned, the resulting statement amounts to a formula with existential quantifiers for the unmentioned entities. HighLife devised clause systems specifically for this case, used Weighted MaxSat reasoning over all the partial observations, and could thus infer non-binary statements relating more than two entities.

8.5.3 Integer Linear Programming

Another way of operationalizing the reasoning over uncertain statements and consistency constraints is by means of *integer linear programs (ILP)* (see, e.g., [516]). These models have been extensively studied for all kinds of industrial optimization problems, such as production planning for factories, supply chains and logistics, or scheduling for airlines, public transportation, and many other applications. ILP is a very mature methodology, hence a candidate for our setting. In general, an ILP consists of

- a set of decision variables, often written as a vector x, allowed to take only non-negative integer values,
- an objective function $c^T \cdot x$ to be maximized, with a vector c of constants, and
- a set of inequality constraints over the decision variables, written in matrix form as $Ax \leq b$ with matrix A and vector b holding constants as coefficients.

To map our consistency reasoning problem onto an ILP, we associate each uncertain statement $A_i = \langle \texttt{SPO} \rangle$, that is, a logical atom, with a decision variable X_i. Their Boolean nature, deciding on whether to accept a statement or not, is realized by restricting X_i to be a **0-1 variable**, by adding constraints $X_i \leq 1$ and $X_i \geq 0$. The clauses that we construct by grounding logical constraints (as explained in Subsection 8.5.2) are encoded into a set of inequalities that couple the decision variables. Weights of candidate statements become coefficients for the ILP objective function, and weights for grounded constraints become the coefficients for a big inequality system.

ILP for Constraint Reasoning:
Input:

- Uncertain candidate statements $A_1 \dots A_n$, each a logical atom of the form $\langle \texttt{SPO} \rangle$ or $\neg\langle \texttt{SPO} \rangle$. The weight of A_i is w_i.
- A set of clauses with more than one atom, one clause for each grounded constraint: $C_1 \dots C_m$ where each C_j consists of positive atoms (without $\neg$) and negative atoms (with $\neg$). We denote these subsets of atoms as C_j^+ and C_j^-. The weight of C_j is u_j.

Construction of the ILP:

- For each statement A_i, there is a 0-1 decision variable X_i.
- The objective function of the ILP is to maximize $\sum_{i=1..n} w_i X_i$.
- For each grounded constraint C_j we create an inequality constraint:
  $\sum_{\mu \in C_j^+} X_\mu \ + \ \sum_{\nu \in C_j^-} (1 - X_\nu) \ \geq \ 1$
  which is equivalent to the condition that at least one atom in the clause should be satisfied (taking the polarities, positive or negative, into account).

The above ILP enforces all grounded constraints to hold. In other words, it does not consider the weights of the original constraints, which would allow slack for exceptions. To achieve this interpretation of *soft constraints*, we need to extend the objective function. Essentially, we add a cost term, proportional to the constraint weight, each time we violate a grounded constraint. The overall objective function would then take this form:

$$\text{maximize } \lambda \ \textstyle\sum_{i=1..n} w_i X_i$$
$$- \ (1-\lambda) \ \textstyle\sum_{j=1..m} u_j \left(\sum_{\mu \in C_j^+} (1 - X_\mu) \ + \ \sum_{\nu \in C_j^-} X_\nu \right)$$

where μ and ν range over the subscripts of the respective atoms, and λ is a tunable hyper-parameter. The sum over statements is the *benefit* from accepting many candidates, and the sum over grounded constraints is the *penalty* to be paid for constraint violations. This basic form can be varied in other ways.

The outlined approach has shown how to incorporate soft constraints, whereas our original ILP formulation allowed only hard constraints. By combining both, via inequality constraints as well as penalty terms in the objective function, we have a choice about which of the original consistency constraints should be treated as strict invariants and which ones can be treated in soft form with allowance for exceptions.

Computing the optimal solution for an ILP is, not surprisingly, also NP-hard. On the positive side, ILPs are very versatile and widely used in all kinds of mathematical optimizations; therefore, a vast array of algorithms exist for making ILPs tractable in many practical cases. Also, there are mature software packages that are very well engineered, most notably, the Gurobi solver (https://www.gurobi.com/). A common optimization technique is to *relax* the ILP by dropping the requirement that the variables can take only integer values. Instead we allow real-valued solutions between 0 and 1, this way treating the ILP as a standard linear program (LP). This technique is known as the **LP relaxation** of the ILP. LPs can be tackled much more efficiently than ILPs, with polynomial algorithms and various accelerations. To obtain a valid solution for the ILP, the solution must be rounded to one of the neighboring integers (e.g., 0.65 can be rounded to 1 or 0). A principled method is to *randomly round* by tossing a coin that falls on 1 with a probability proportional to the real value (e.g., with probability 0.65 becoming 1 and with probability 0.35 becoming 0). This randomized algorithm has good approximation guarantees with high probability [404, 592].

ILP models have been applied to various tasks of KB cleaning as well as for the underlying knowledge extraction. [500, 498] provide general discussion on using ILP for KB curation and for NLP tasks.

8.5.4 Probabilistic Graphical Models

The prior uncertainty in the weighted MaxSat problem can also be modeled in a probabilistic fashion, such that the Boolean decision variables become *random variables* with probabilities for being true or false. This line of models often starts with a *declarative specification* of candidate statements and logical constraints, just like we did in the previous subsections on MaxSat and ILP. For probabilistic inference, the high-level specification is translated into a **Markov Random Field (MRF)**, the general framework for probabilistic graphs that couple a large number of random variables (see, e.g., [283]). Conditional Random Fields (CRF) that are widely used for entity discovery (Chapter 4) also fall into this regime (see Section 4.4.1). For tractability, these models have to make assumptions about conditional independence, thus becoming **probabilistic factor graphs** with factors expressing the local coupling of (small) subsets of (non-independent) random variables.

Markov Logic Networks:
In the following, we focus on one prominent and powerful model from this broad family: *Markov Logic Networks (MLN)* [125].

Markov Logic Network (MLN):
Input:

- a set of grounded clauses $C_1, ..., C_n$, derived from uncertain statements and soft constraints, with weights $w_1, ..., w_n$, and
- binary random variables $X_1, ..., X_k$ for the atoms that appear in the clauses.

Construction of the MLN:
The corresponding MLN is an undirected graph with random variables $X_1 \dots X_k$ as nodes, and edges between nodes X_i, X_j if these variables are considered to be coupled. Conversely, if we assume conditional independence

$$P[X_i \mid X_j, \text{all } X_\nu \neq X_i \text{ and } \neq X_j]$$
$$= P[X_i \mid \text{all } X_\nu \neq X_i \text{ and } \neq X_j]$$

then there is no edge between X_i and X_j. Otherwise, X_i and X_j are coupled by an edge between them.
The MLN constitutes a joint probability distribution for the variables $X_1 \dots X_k$, further explained below.

As an example, reconsider the set of 8 clauses over 5 atoms discussed in Section 8.5.2. To make it more interesting, let us add another constraint that leads to a 9th clause:

$\neg\langle$`BobDylan hasWon Grammy`$\rangle$
$\vee \neg\langle$`BobDylan hasWon LiteratureNobelPrize`$\rangle$
$\vee \neg\langle$`BobDylan type scientist`$\rangle$ [0.6]

(if someone has won a Grammy and a Literature Nobel Prize, she/he cannot be a scientist). The constraints are the mechanism for coupling the random variables for the candidate statements; variables that do not share any constraints are conditionally independent. This leads to the grounded MLN in Figure 8.3. The upper part shows the actual graph with binary edges; the lower part shows the same graph with cliques in the graph made explicit by the small blue-rectangle connectors. So this MLN of five atoms has three cliques of size 2 and one clique of size 3. We added the new constraint for this very illustration of a clique with more than two nodes.

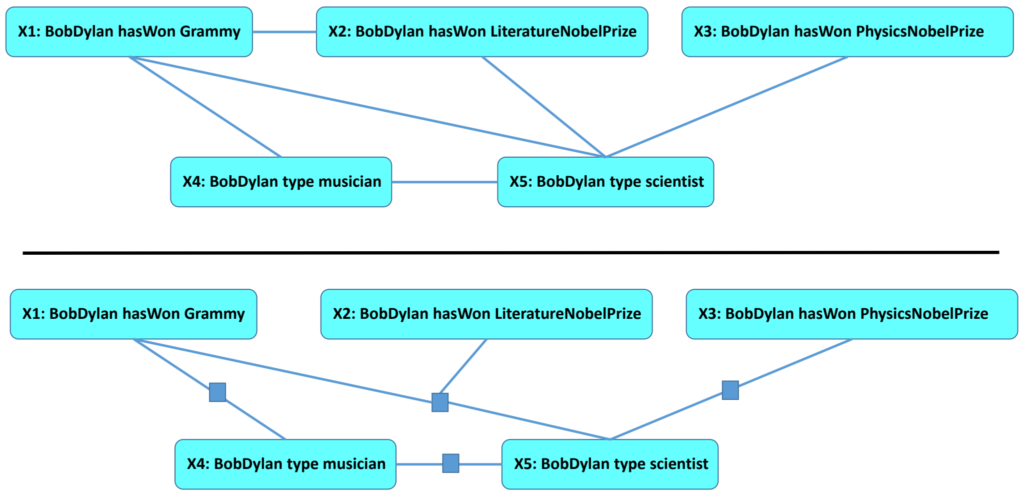


Figure 8.3: Example for MLN Graph

Factorized Distribution of MLN:
The MLN graph induces a *joint probability distribution* for all random variables together. By the assumptions about which variables are coupled and which ones are conditionally independent, and by the Hammersley-Clifford Theorem from MRF theory [283, 125], the joint distribution takes the following factorized form:

$P[X_1 \dots X_k] \sim \prod_{cliques\ j} \Phi_j\ (X_{j_1} \dots X_{j_l}$ forming clique $j)$

where j ranges over all cliques in the graph and Φ_j are so-called *clique potential functions* for capturing local probabilities. These per-clique terms are the *factors* of the **factor graph**.
For our KB curation setting, the cliques correspond to the constraint clauses. This leads to the product form

$P[X_1 \dots X_k] = \frac{1}{Z} \prod_{C_j}\ e^{w_j}$ with C_j ranging over all clauses satisfied by $X_1 \dots X_k$

with clause weights w_j and normalization constant Z.

An MLN defines a probability distribution over all *possible worlds*, that is, over all possible joint assignments of truth values to the variables $X_1, ..., X_k$. Computing the world with the maximum joint probability is equivalent to solving a weighted MaxSat problem. Finding this world is known as **MAP inference**, where MAP stands for *maximum a* *posteriori*. Like MaxSat, this computation is NP-hard (or, more precisely, #P-complete, a related and potentially harder complexity class for counting problems) [497]. Practical methods for MAP inference, therefore, resort to **Monte Carlo sampling**, most notably, *Gibbs sampling*, or to variational calculus (see also Section 4.4.1 on CRFs). Approximation algorithms for MaxSat and ILP have been used as well (see, e.g., [490]). In addition to the joint MAP, MLN inference can also compute the *marginal probability* for each variable which can be interpreted as the confidence in an individual statement being valid. Unfortunately, this task comes at even higher computational complexity than MAP, and is hardly supported by any software tools for probabilistic graphical models.

The weights for the MLN clauses can be automatically learned from training data, without reliance on human inputs or external statistics. However, this learning is itself an expensive task, as it involves non-convex optimization. It is typically addressed by gradient descent methods (see also Section 4.4.1).

MLNs have been used for a variety of probabilistic reasoning tasks [489, 125, 126], including entity linking (e.g., [537]) and the extraction of is-a and part-of relationships from text sources (e.g., [458]). Also, advanced methods for *minimum-repair database cleaning* have made clever use of MLN inference [486].

A prominent case of using MLN models for KB construction is the **DeepDive** project (http://deepdive.stanford.edu/) [527, 527, 655, 656]. It comprises a framework and software suite for constructing KBs from scratch as well as augmenting them in an incremental manner. DeepDive has been applied to build domain-specific KBs, for example, for paleobiology, geology and crime fighting (fighting human trafficking [279]). Usability for knowledge engineers is boosted by having a declarative interface for inputs, with automatic translation into MLNs and, ultimately, MRFs. Nevertheless, high-quality KB construction requires substantial care and effort for proper configuration, training, tuning and other human intervention [656]. Like with MaxSat, a major concern for scalability is that the grounding step – moving from logical constraints with quantified variables to fully instantiated clauses – comes with the risk of combinatorial explosion. To mitigate this issue, the DeepDive project has developed effective techniques for **lazy grounding**, avoiding unnecessary instantiations [428].

Another prominent system that makes use of MLNs is **StatSnowball** [672, 425]. This system uses both predefined MLN constraints and logical patterns that are learned at run-time. The former express prior knowledge, such as $hasMother(x, y) \Rightarrow hasChild(y, x)$. The learned kind of soft invariants, on the other hand, capture the relationship between textual patterns and the relations and their arguments, similar to the statement-pattern duality in DIPRE [58] (see Section 4.3.2) and the reasoning for extraction in SOFIE [566] (see Section 8.5.2).

Probabilistic Soft Logic (PSL):
MLNs relax the Weighted MaxSat model by admitting many sub-optimal worlds with lower probabilities. In each of these worlds, a Boolean variable is true or false, with certain probabilities. This setting can be relaxed further by allowing different *degrees of belief* in a variable being true [23]. Essentially, the discrete optimization problem of MaxSat and MLNs is relaxed into a continuous optimization by making all random variables real-valued. This technique is analogous to relaxing ILPs into LPs (see Section 8.5.3). Note that these real-valued degrees of truth are different from the probabilities (which are real-valued anyway, for discrete and continuous models alike). Every combination of real-valued belief degrees, on a continuous scale from 0 to 1, is associated with a probability density.

For making this approach tractable, the truth value, or *interpretation* I, of a conjunction of variables is defined as the Lukasiewicz t-norm: $I(v \wedge w) = max\ (0,\ I(v) + I(w) - 1)$. Each clause $a_1 \vee ... \vee a_n$ is treated as a rule $\neg a_1 \wedge ... \wedge \neg a_{n-1} \Rightarrow a_n$, with body $\neg a_1 \wedge ... \wedge \neg a_{n-1}$ and head a_n (see Section 8.3.3). The *distance from satisfaction* of this rule in an interpretation I is defined as $d_I(body \Rightarrow head) = max\ (0,\ I(body) - I(head))$. This yields the following setting:

Probabilistic Soft Logic (PSL) Program:
For a set of rules $r_1, ..., r_n$ (including atoms for statement candidates) with weights $w_1, ..., w_n$, a PSL program computes the probability distribution over interpretations I:

$$P[I] = \frac{1}{Z} exp(-\sum_{i=1..n} w_i d_I(r_i))$$

where Z is a normalizing constant.

Like for MLNs, the central task in PSL is MAP inference, for the value assignment (degrees of being true) that maximizes the joint probability over all variables together. [23] has shown that, unlike for discrete MLNs, this can be computed in polynomial time. By using a Hinge-loss objective function, MAP inference becomes a convex optimization problem – still not exactly fast and not easily scalable, but no longer NP-hard. PSL has been used for a variety of tasks, from entity linking [290] and paraphrase learning [192] to KB cleaning [464].

Further Works on Probabilistic Factor Graphs:
Numerous other works have leveraged variants of probabilistic graphical models for different aspects of KB construction and curation. These include *constrained conditional models* by [81], *factor graphs* for joint extraction of entities and relations [491, 643] (see also Section 6.2.2.4), *coupled learners* for the NELL project [394, 395] (see also Section 9.3), and others.

8.6 KB Life-Cycle

Construction and curation of a knowledge base is a never-ending task. As the world keeps evolving, knowledge acquisition and cleaning needs to follow these changes, to ensure the freshness, quality and utility of the KB. This life-cycle management, over years and decades, entails several challenges. Surprisingly, these issues have received little attention in the KB research community. There are great opportunities for novel and impactful work.

8.6.1 Provenance Tracking

While provenance of data is a well-recognized concern for database systems, with long-standing research (e.g., [63, 64]), it is largely underrated and much less explored for knowledge bases. In a KB, each statement should be annotated with **provenance metadata** about:

- the **source** (e.g., web page) from where the statement was obtained, or sources when multiple inputs are combined,
- the **timestamp(s)** of when the statement was acquired, and
- the **extraction method(s)** by which it was acquired, for example, the rule(s), pattern(s) or classifier(s) used.

This is the minimum information that a high-quality KB should capture for manageability (see, e.g., [71, 240]). In addition, for learning-based extractions, the underlying *configuration* (e.g., hyper-parameters) and *training data* should be documented. For human contributions, the assessment and approval steps, involving *moderators/curators*, needs to be documented (see, e.g., [452, 453]).

This metadata is crucial for being able to trace errors back to their root cause, when they show up later: the spurious statements, the underlying sources and the extraction methods. This way, provenance tracking supports removing incorrect KB content or correcting the errors. For query processing, provenance information can be propagated through query operators. This allows tracing each query result back to the involved sources [63], an important element of explaining answers to users in a human-comprehensible way.

Provenance information can be added as additional arguments to relational statements, in the style of this example:

⟨`Bob Dylan, won, Nobel Prize in Literature,`
`source: www.nobelprize.org/prizes/literature/2016/dylan/facts,`
`extractor:lstm123`⟩,

or by means of reification and composite objects in RDF format (see Section 2.1.3). Another option is to group an entire set of statements into a **Named Graph** [213], a W3C-approved way of considering a set of statements as an entity [211, 73]. Then, provenance statements can be made about such a group entity. Yet another option is RDF* [212, 214], a proposed mechanism for making statements about other statements. The most common solution, however, is what is known as **Quads**: triples that have an additional component that serves as an identifier of the triple. Then, provenance statements can be attached to the identifier. Conceptually, this technique corresponds to a named graph of exactly one statement; it is used, for example, in YAGO 2 [240] and in Wikidata Various triple stores support quads.

8.6.2 Versioning and Temporal Scoping

Nothing lasts forever: people get divorced and marry again, and even capitals of countries change once in a while. For example, Germany had Bonn as its capital, not Berlin, from 1949 to 1990. Therefore, *versioning* and *temporal scoping* of statements are crucial for proper interpretation of KB contents. In addition, versioning rather than over-writing statements is also useful for quality assurance and long-term maintenance. For database systems this is an obvious issue, and industrial KBs have certainly taken this into account as well. However, academic research has often treated KBs as a one-time construction effort, disregarding their long-term life-cycle.

Versioning:
Keeping all versions of statements, as relationships between entities are established and dissolved, supports *time-travel search*: querying for knowledge as of a given point in the past (e.g., for Germany's capital in 1986). KB projects like DBpedia and YAGO approximated version support by periodically releasing new versions of the entire KB. In addition, YAGO 2 [240] introduced systematic annotation with temporal scopes to many of its statements (see below). The Wikidata KB keeps histories of individual items (e.g., https://www.wikidata.org/w/index.php?title=Q392&action=history about Bob Dylan) and supports convenient access to earlier versions on specific entities. Also, SPARQL queries can be performed over the Wikidata edit history [580].

Temporal Scopes:
We aim for temporally scoped statements, by annotating SPO triples with their **validity times**, which can be timepoints or time intervals. This can be expressed either via higher-arity relations, such as

```
wonPrize (EnnioMorricone, Grammy, 11-February-2007)
wonPrize (EnnioMorricone, Grammy, 8-February-2009)
capital (Germany, Bonn, [1949-1990])
capital (Germany, Berlin, [1991-now])
```

or by means of reification and composite objects (see Section 2.1.3). Note that the temporal scopes can have different granularities: exact date, month and year, or only the year. The choice depends on the nature of the event or temporal fact, and also on the precision of how they are reported in content sources.

Discovering and Inferring Temporal Scopes:
Given a statement, such as `married (Bob Dylan, Sara Lownds)`, how can we determine the begin and end of its validity interval, and how do we go about assigning timepoints to events? There is a variety of methods for spotting and normalizing **temporal expressions** in text and semi-structured content, based on rules, patterns or CRF/LSTM-like learning – see, for example, [107, 597, 166, 240, 302, 555]. Methods for property extraction (see Chapter 6) can be applied to linking these time points or intervals to respective entities, and thus assigning them to SPO triples, at least tentatively. In addition, special pages, categories and lists in Wikipedia, such as monthly and daily events (e.g., https://en.wikipedia.org/wiki/Portal:Current_events/2020_July_14) and annual chronologies (e.g., https://en.wikipedia.org/wiki/2020) provide great mileage [303, 302, 530].

The resulting extractions of temporally scoped statements may result in noisy and conflicting outputs, though. For example, we may obtain statement candidates:

```
married (Charles, Diana, [1981-1996]): [0.7]
married (Dodi, Diana, [1995-1997]): [0.6]
married (Charles, Camilla, [1990-2020]): [0.5]
married (Andrew, Camilla, [1973-1997]): [0.4]
```

where the numbers in brackets, following the statements, denote confidence scores. Together, these statements imply some cases of illegal bigamy: being married to more than one spouse at the same time. This is again a case for imposing consistency constraints and reasoning to clean this space of candidates. We can use the repertoire from Section 8.5, including MaxSat, integer linear programming (ILP), or probabilistic graphical models. For example, the cleaning task could be cast into an ILP as follows:

ILP for Temporal Scoping:
Input: Candidates of the form $\langle S, P, O, T \rangle : w$,
with time interval T and confidence score w
Decision Variables:

- $X_i = 1$ if candidate i is accepted, 0 otherwise
- $P_{ij} = 1$ if candidate i should be ordered before j

Objective Function: maximize $\sum_i w_i \cdot X_i$
Constraints:

- $X_i + X_j \leq 1$ if candidates i and j overlap in time and are conflicting (e.g., violating the monogamy law)
- $P_{ij} + P_{ji} \leq 1$ for all i, j, for acyclic ordering
- $(1 - P_{ij}) + (1 - P_{jk}) \geq (1 - P_{ik})$ for all i, j, k, for transitivity if i, j, k must be ordered
- $(1 - X_i) + (1 - X_j) + 1 \geq (1 - P_{ij}) + (1 - P_{ji})$ for all i, j that must be totally ordered – in order to couple the X_i and P_{ij} variables

This ILP model can be seen as a template for all kinds of temporal-scope constraint reasoning, generalizing the anti-bigamy case at hand.

Unfortunately, by the temporal overlap of our four candidate statements, this would conservatively accept only the first statement about Charles and Diana being married from 1981 through 1996 and the last statement about Andrew and Camilla. A technique to improve the recall of such reasoning is to decompose the temporal scopes of the candidates into disjoint time intervals, cloning the non-temporal parts of the statements. For example, the marriage of Charles and Camilla is split into one statement for [1990,1996] and another one for [1997,2020]. This would allow the reasoner to reject the scope [1990,1996] while accepting [1997,2020].

Such techniques and the application of consistency reasoning for temporal scoping of KB statements have been investigated by [345, 573, 572, 615, 614].

A variation of this theme is to discover and infer the **relative temporal ordering** between events and/or the validity of statements. This aims to detect relationships like `happened-before`, `happened-after`, `happened-during` etc. Methods along this lines include [590, 271, 392, 427].

8.6.3 Emerging Entities

An important aspect of evolving knowledge is to cope with newly **emerging entities**. When discovering entity names in Web sources, we aim to disambiguate them onto already known entities in the KB; see Chapter 5. However, even if there is a good match in the KB, it is not necessarily the proper interpretation. For entity linking, this is the **out-of-KB entity** problem [343].

For example, when the documentary movie *"Amy"* was first mentioned a few years ago, it would have been tempting to link the name to the soul singer `Amy Winehouse`. However, although the movie is about the singer's life, the two entities must not be confused. Today, after having won an Oscar, the movie is, of course, a registered entity in all major KBs. The general situation, though, is that there will be a delay between new entities coming into existence and becoming notable. Likewise, existing entities that are not notable enough to be covered by a KB may become prominent overnight, such as indie musicians getting popular or startup companies getting successful. Recognizing these as early as possible can accelerate the KB growth, and most importantly, it is crucial to avoid confusing them with pre-existing entities that have the same or similar names.

To address the problem, the methods for entity linking (EL) (see Chapter 5) always have to consider an additional candidate `out-of-KB` for the mapping of an observed mention. Whenever the EL method has higher confidence in this choice than in any candidate entity from the KB, the mention should be flagged accordingly. As confidence scores are not always well-calibrated, more sophisticated scoring and score-calibration methods have been investigated [204, 236].

These techniques are conservative, in that they avoid incorrectly mapping an emerging entity onto a pre-existing one. However, this alone is not sufficient, because we do want to add the emerging entity to the KB at some point. Moreover, there could be multiple out-of-KB entities with the same or similar names in the discovered text mentions. For example, in addition to the movie "Amy" (about Amy Winehouse), the character "Amy" (Farrah Fowler) from the TV series "Big Bang Theory" could also become a candidate for addition to the KB.

An approach to handle such cases has been proposed in [237]. For each mention mapped to `out-of-KB`, a *contextual profile* is created and maintained. This comprises the mention itself and keyphrases from the surrounding contexts or latent models derived from contexts (cf. Section 5.3.2). The profile is gradually enhanced as we observe more mentions of what is likely the same out-of-the-KB entity. After a while, we obtain a repository of emerging entity names with their contextual profiles. When the profile of a name is rich enough to infer its semantic type(s), such as `documentaryMovie` or `fictitiousCharacter`, we may consider adding the emerging entity into the KB, properly registered with its type(s).

There are still two caveats to consider. First, although there is initial evidence for an emerging out-of-KB entity, it may turn out later that this actually denotes an already known entity in the KB. So the method has to periodically reconsider and possibly revise the EL decision. Second, as the same name may denote multiple out-of-KB entities, the contextual profile for this name could improperly conflate more than one emerging entity. The "Amy" scenario is an example. To handle such cases, the method needs to consider splitting a profile, to identify strongly coherent contexts – one for each emerging entity. In the long run, human-in-the-loop curation may still be required for keeping the KB at its high quality.

Further methods for early discovery of emerging entities have been developed, for example, by [261, 647, 6, 435, 661], with emphasis on social media or web tables as sources. For entities of type `event`, news digests such as daily pages in Wikipedia (e.g., https://en.wikipedia.org/wiki/Portal:Current_events/2020_July_14) are valuable assets, too. Note that these daily pages often contain headlines that are not yet covered by any of the encyclopedic articles in Wikipedia.

Finally, a methodological related issue is to correct existing KB statements where the object is simply a string literal, such as "Amy – Oscar-winning documentary movie, 2015" in a triple like ⟨`AmyWinehouse featuredIn` "Amy . . . 2015"⟩. This is a frequently arising case as the initial knowledge acquisition may only pick up strings but miss out on proper entity linking. [84] has developed a general framework for correcting these omissions, utilizing EL techniques and consistency constraints.

8.7 Take-Home Lessons

We highlighted that knowledge bases require continuous curation, from quality assessment to quality assurance. Key lessons are the following:

- Quality measures for correctness and coverage are computed by *sampling* statements with human judgements, by crowdsourcing or, if necessary, experts.
- No KB is ever complete: assessing and predicting *completeness* often builds on the Local Completeness Assumption (LCA).
- Logical *invariants* about the KB content serve twofold roles: as *constraints* they can detect erroneous statements to keep the KB consistent; as *rules* they can deduce additional statements to fill gaps in the KB.
- Logical patterns in the KB can be automatically discovered, to *learn rules* and to *analyze bias*.
- For cleaning candidate statements at scale, *constraint-based reasoning* is a best-practice approach, with a suite of models and methods from MaxSat and ILP to probabilistic factor graphs.
- For the long-term *life-cycle* of a KB, tracking the provenance of statements as well as versioning with temporal scopes are essential components.

9 Case Studies

9.1 YAGO

The YAGO project (https://yago-knowledge.org, starting in 2006) created the first large knowledge base that was automatically extracted from Wikipedia, largely in parallel to DBpedia, discussed in Section 9.2. YAGO has been maintained and advanced by the Max Planck Institute for Informatics in Germany and Télécom Paris University in France. The KB was used in many projects world-wide, most notably, for semantic type checking in the IBM Watson system that won the Jeopardy quiz show [163, 409].

9.1.1 Design Principles and History

Initial YAGO: Core Knowledge
The key observation was that Wikipedia contains a large number of individual entities, such as singers, movies or cities, but does not organize them in a semantically clean type system. Wikipedia's hierarchy of categories was not suitable as a taxonomy. WordNet, on the other hand, has a very rich and elaborate taxonomy, but is hardly populated with instances. YAGO aimed to combine the two resources to get the best of both worlds.

The first version of YAGO [562] converted every Wikipedia article into an entity, and extracted its classes from the categories of the article. To distinguish between thematic categories for human browsing (e.g., *Rock 'n Roll music* for Elvis Presley) and properly taxonomic categories (e.g., *American singers*), YAGO developed the heuristics discussed in Section 3.2. If the head noun of a category name is in plural form (as in *American singers*), then it is a taxonomic class. These judiciously selected leaf-level categories were linked to WordNet with the methodology presented in Chapter 3.

The first version of YAGO also extracted selected kinds of facts from Wikipedia categories. A small set of relations was manually identified, including (*hasWonAward*, *isLocatedIn*, etc.), and regular expressions were specified for the corresponding Wikipedia categories (e.g., *Grammy Award winners* or *Cities in France*). YAGO extracted labels for entities from Wikipedia redirects, and attached provenance information to each statement. The hand-crafted specification included domain and range constraints to eliminate spurious statements. The objective was to focus YAGO entirely on precision, even if this meant a loss in recall. The rationale was that a KB with 5 million facts and 95% precision is more useful than a KB with 10 million facts and 80% precision.

The quality of the extracted statements was evaluated manually by a sampling technique. A random sample was drawn for each relation, and the statements were manually compared to Wikipedia as ground truth. The number of samples was chosen so as to bring the Wilson confidence interval of the estimated precision to 95% $\pm$ 5%. Refining by relation, some properties including the `type` property even reached nearly 99% precision [563, 240]. For a long time, YAGO was the only major KB that came with such statistical guarantees about its correctness.

In 2008, YAGO was extended with facts from infoboxes of Wikipedia [563]. These were extracted by hand-crafted regular expressions (see Section 6.2.1) for around 100 selected relations. This process also extracted the validity times of statements when applicable. The SPO triple notation was slightly extended to allow statements with temporal scope, such as:

```
Germany hasGDP "$3,667 trillion" inYear "2008"
```

By reification (see Section 2.1.3), this shorthand notation was mapped into pure triples (or quads, see Section 8.6.1).

The data model was extended by a taxonomy of types for literal values. For example, the literal “$3,667 trillion” was linked by the property `hasValue` to “3,667,000,000,000”, which is an instance of the class `integer`, a subclass of `number`, which is a subclass of `literal`. Likewise, the property `hasUnit` could capture the proper currency US dollars. This further strengthened the ability for early type checking and ensuring near-human quality.

The YAGO KB was the focal point of the broader *YAGO-NAGA* project that included methods and tools for exploring and searching the KB [278].

YAGO 2: Spatial and Temporal Scoping

In 2010, YAGO was systematically extended with temporal and spatial knowledge [239, 240]. Entities and statements had time intervals assigned to denote when entities existed and statements were valid. Timestamps were extracted from the Wikipedia infoboxes when possible, and propagated to other statements and entities by a limited form of Horn rules (see Section 8.3.2). For example, if we know the birth date and the death date of Frank Sinatra, then we can deduce the validity interval for the fact that he was a person. Such rules were systematically applied to people, artifacts, events, and organizations, thus giving validity times to about one third of the facts. This temporal scoping was in turn beneficial as a consistency check when extending the KB or building a new major version.

The spatial dimension of YAGO came from GeoNames https://www.geonames.org/), a large repository of geographical entities, with coordinates and informative types. To avoid duplicates, the entities from Wikipedia were matched to the entities in GeoNames by comparing names and geographical coordinates with thresholds on similarity. This simple entity-matching technique (cf. Section 5.2) preserved the high quality of canonicalized entities. The type taxonomy of GeoNames was mapped to the class taxonomy of YAGO, by taking into account the name of the GeoNames class, the head noun of that name, the most frequent meaning in WordNet, and the overlap of the glosses in the two resources. Similar to the temporal scoping, Horn rules were applied to propagate locations from entities to their statements, and vice versa.

This KB was accompanied by a query engine [239, 240] that allowed searching entities not just by their facts, but also by validity times, spatial proximity and space-time combinations. For example, the query

GeorgeHarrison created ?s after JohnLennon

would find songs written by George Harrison after John Lennon's death, and the query

guitarists bornIn ?p near Seattle

would return Jimi Hendrix, Kurt Cobain, Carla Torgerson and more. Recent work on augmenting YAGO2 with spatial knowledge is the *Yago2geo* project [276], which integrated content from OpenStreetMap and other sources, and supports a very expressive query language called GeoSPARQL.

The extraction patterns for YAGO2 were specified declaratively. An example pattern is:

```
"Category:(.+) births" pattern "\$0 wasBornOnDate Date(\$1)"
```

to extract birth dates from Wikipedia category names. The next version of YAGO, YAGO2s [43] further advanced the principle of declarative specifiction and modularization. The system was factored into 30 *extractors*, each for a specific scope. The extractors were orchestrated by a dependency graph, where each module depends on inputs from other modules. A scheduler could run these extractors largely in parallel or in pipelined mode. This DB-engine-like declarative machinery proved very valuable for debugging and quality assurance, and could build new KB versions very efficiently (cf. Section 6.2.1.4).

YAGO 3: Multilingual Knowledge

In 2014, YAGO 3 [364] started extracting from Wikipedia editions beyond English, covering editions like German, French, Dutch, Italian, Spanish, Romanian, Polish, Arabic, and Farsi (based on the authors' language skills, for validation of results). The goal was to construct a single, consolidated KB from these multilingual sources, with more entities and statements (as many appear only in specific editions) but without any duplicates. The extractors harnessed inter-wiki links between Wikipedia editions and the inter-language links in Wikidata (https://wikidata.org/). For example, the French Wikipedia article about *Londres* is about the same entity as the English article about *London*.

A difficulty to address was the extraction from non-English infoboxes, without manually specifying patterns for each different edition. For this purpose, the YAGO 3 system employed distant supervision from the English edition (see Sections 6.3 and 7.3.2). By statistically comparing English seeds for subject-object pairs against those observed in non-English infoboxes, the system could learn the correspondences between properties from different sources.

To construct the taxonomy, the foreign category names were mapped to their English counterparts, by harnessing inter-language links from Wikidata. The modular architecture developed for YAGO2s allowed all this with just a handful of new extractors: translation of non-English entities, mapping of infobox attributes to relations, and taxonomy construction. Downstream extractors were not affected by these extensions.

Another major project on multilingual knowledge, with even larger coverage, is BabelNet [418, 145] (https://babelnet.org/).

YAGO 4: Alignment with Wikidata

YAGO 3 has been continuously improved, and the software became open source [483]. However, the KB was bound to Wikipedia and the entities featured there. Thus, it became clear that the KB could never reach the scale of entity coverage that Wikidata achieved in the meantime, with nearly 100 million entities. On the other hand, Wikidata has the principle of including claims as statements (see Section 9.4, with potential diversity of perspectives, rather than undisputed factual statements only. Therefore, semantic constraints cannot be rigorously enforced. Furthermore, the large number of contributors to the Wikidata community has led to a convoluted and cluttered taxonomy of classes, where an entity such as *Paris* is buried under 60, mostly uninformative, classes, 20 of which are called "object", "unit", "seat", "whole", etc.

The cleaning of contradictory statements and the transformation of convoluted and noisy taxonomies into clean type systems have been key competences of the YAGO project from its very start. In this spirit, the latest YAGO version, YAGO 4 [582], abandoned Wikipedia as input source, and set out to tap into Wikidata as a premium source, applying the same principles (cf. Chapter 3). The higher-level types of the taxonomy are no longer based on WordNet; instead YAGO 4 adopts the type system of *schema.org* [193], an industry standard for semantic markup in web pages.

The directly populated leaf-level classes of YAGO 4 are carried over from Wikidata. In this regard, Wikidata is fairly clean and coarse-grained; most entities belong to only one or two types directly. For example, all people have type `human` and none of the conceivable fine-grained types such as singer, guitarist etc. Wikidata expresses the latter by means of various properties. Since schema.org has only ca. 1000 classes and the overlap with the immediate types of Wikidata instances is even smaller, it was best to manually align the relevant types from the two sources. Properties are also adopted from schema.org, with the advantage that they come with clean type signatures for domain and range. Again, the alignment with Wikidata properties required a reasonably limited amount of manual work. The entities, Wikidata's best asset, were transferred from Wikidata to populate the newly crafted KB schema. This data was complemented by hand-crafted consistency constraints for class disjointness, functional dependencies and inclusion dependencies (see Section 8.3), expressed in the SHACL language (see Section 8.3.1).

The resulting KB comprises ca. 60 million entities with 2 billion statements, organized into a clean and logically consistent taxonomic backbone. For this high quality, the KB construction "sacrificed" about 30 million Wikidata entities that had to be omitted for consistency. However, these affect only the long tail of less notable entities which have very few properties. As a result of this constraint-aware construction process, the YAGO 4 knowledge base is "reason-able" [582]: provably consistent and amenable to OWL reasoners. Metadata, about provenance, is represented in the RDF* format [212].

9.1.2 Lessons Learned

YAGO was one of the first large KBs automatically constructed from web sources. Its unique traits are high precision, semantic constraints, and a judiciously constructed and fairly comprehensive type taxonomy. Over YAGO's 15-year history, several major lessons were learned:

- **Harvest low-hanging fruit first:** YAGO was successful because it focused on premium sources that were comparatively easy to harvest and could yield high-quality output: the category system of Wikipedia and semi-structured content like infoboxes. This resulted in near-human precision that was previously unrivaled by automatic methods for information extraction at this scale.

- **Focus on precision:** YAGO has focused on precision at the expense of recall. The rationale is that every KB is incomplete, and that applications are thus necessarily prepared to receive incomplete information. Under this regime, a user is not surprised if some song or city is missing from the KB. Conversely, users are confused or irritated when they encounter wrong statements. Therefore, YAGO focused on premium sources and relatively conservative extration methods, reserving more aggressive methods for KB augmentation. The project investigated and developed a variety of such advanced methods as well, including the SOFIE tool [566] (see Section 8.5.2), but these had limited impact on the more conservative releases of the YAGO KB. Nevertheless, SOFIE and its scalable parallelization [412] were successfully used in another project on building a health KB called *KnowLife* [150, 148].

- **Limited-effort manual contributions:** The project identified sweet spots where limited manual effort had a very large positive impact. The specification of properties and their type signatures is a case in point. It is not much effort to define hundreds of properties and constraints by hand: a small price when these can guarantee the cleanliness and tangiblity of many million statements.

- **Modularized extractors:** There is enormous value in unbundling the KB construction code into smaller extractor modules, each with a specific scope. This allows declarative orchestration of an entire extractor ensemble, and greatly simplifies maintenance, debugging, and project life-cycle.

- **Use open standards:** The adoption of open standards (like RDF, RDFS, RDF* [212], and SHACL) boosts the usage and utility of KB resources and APIs by a wider community of researchers and developers. This includes the representation of statements. In the early years of YAGO, many questions arose about the syntax of KB statements, character encodings, reserved characters, escape conventions, etc. These could have been avoided by an earlier full adoption of RDF.

9.2 DBpedia

DBpedia (https://dbpedia.org, starting in 2007) was the other early project to construct a large-scale knowledge base from Wikipedia contents [18, 19, 317]. DBpedia spearheaded the idea of *Linked Open Data (LOD)* [225]: a network of data and knowledge bases in which equivalent entities are interlinked by the *sameAs* predicate. The LOD ecosystem has grown to ten thousands of sources, with DBpedia as a central hub (see also Section 9.6).

DBpedia targeted infobox attributes right from the beginning (cf. Section 6.2.1): every distinct attribute was cast into a property type, without manual curation. Thus, DBpedia could not apply domain and range constraints and other steps for canonicalization and cleaning. In return, DBpedia captured all information of the Wikipedia infoboxes, and thus provided much larger coverage than YAGO. Over the next years [47], the project also extracted further contents; abstracts, images, inter-wiki links between different language editions, redirect labels, category names, geo-coordinates, external links and more, becoming a "Wikipedia in structured form".

In 2009, DBpedia started organizing the entities into a small hand-crafted taxonomy [47], driven by the most frequent infobox templates on Wikipedia. It also specified clean property types for these classes, and manually mapped the infobox templates onto properties. To keep its high coverage, DBpedia stored both the raw infobox attributes (which cover all infoboxes) and the selectively curated ones. All in all, DBpedia offered four different taxonomies: the Wikipedia categories, the YAGO taxonomy, the UMBEL taxonomy (an upper-level ontology derived from Cyc, see https://en.wikipedia.org/wiki/UMBEL), and the hand-crafted DBpedia taxonomy. This shows the difficulty of reaching agreement on a universal class hierarchy. Another novelty in 2009 was "DBpedia live", a system that continuously processes the change logs of Wikipedia and feeds them into the KB as incremental updates [229].

Over the next years, the project built up an international community of contributors [378, 285, 317], and added many useful tools including SPARQL endpoint and other APIs as well as the *Spotlight* tool for named entity recognition and disambiguation [379] (cf. Section 5.4). Separate DBpedia editions were created for each of ca. 100 language editions of Wikipedia. These DBpedias are independent, but volunteers in different countries mapped infobox attributes of non-English Wikipedias to the common DBpedia schema. Finally, DBpedia dealt with the rise of Wikidata (see Section 9.4) by incorporating its entities and properties [259] while keeping its genuine taxonomy. Mappings between Wikidata properties and the DBpedia schema were manually specified by the community.

Since 2014, DBpedia is run by the DBpedia Association, an organization (https://wiki.dbpedia.org/dbpedia-association) with regional chapters in 15 countries.

9.3 Open-Web-based KB: NELL

9.3.1 Design Principles and Methodology

The Never-Ending Language Learner NELL (http://rtw.ml.cmu.edu/, starting in 2010) [71, 72, 394, 395] is a project at Carnegie Mellon University to build a knowledge base "ab initio" from any kinds of web sources. NELL distinguishes itself from other KB projects by its paradigm of *continuously running* over many years, the idea being that the KB is incrementally grown and that the underlying learning-based extractors would gradually improve in both precision and recall. The key principle to tackle this ambitious goal is *coupled learning*: NELL has learners for several tasks, and these tasks are coupled to support each other. For example, learning that *Elvis* is a singer reinforces the confidence in an extraction that Elvis released a certain album, and vice versa. Likewise, learning that *Paris* is the capital of France from a textual pattern and learning it from a web table strengthens the belief that this is a correct fact. This is highly related to harnessing soft constraints for consistency and to the factor coupling of probabilistic graphical models (see Section 8.5).

NELL starts with a manually created schema, with ca. 300 classes and ca. 500 binary relations with type signatures. NELL boostraps its learners with a few tens of labeled training samples for each class and each relation, for example, *guitar* for the class *musicInstruments*, *Colorado* for the class *rivers*, *Page* for the relation *cityLiesOnRiver*, and *released (Elvis, JailhouseRock)* for the relation *released.* The extractors run on a large pre-crawled Web corpus, with the following central learning tasks:

- **Type classification:** Given a noun phrase such as "Rock en Seine" or "Isle of Wight Festival", classify it into one or more of the 300 classes, like `musicfestival` and `event` (cf. NER and entity-name typing in Section 4.4.1). NELL uses different learners for this task, based on: string features (e.g., learning that the suffix "City" in a compound noun phrase often identifies cities, such as "New York City"), textual patterns (such as "mayor of X"), appearance in web tables (e.g., appearance in a column with other entities that are known to be cities), image tags and visual similarities, and embedding vectors.

- **Relation classification:** Given a pair of noun phrases, classify it into one or more of the relations, this way gathering instances of the relations. Again, NELL uses an ensemble of several learners, some of which are similar to the type classifiers. Features include textual patterns, the DOM-tree structure of web pages, and word-level embeddings (cf. Chapter 6).

- **Synonymy detection:** Given a pair of noun phrases, detect whether they denote the same entity (e.g., "Big Apple" and "New York City"). Several supervised classifiers are employed, using features like string similarity or co-occurrence with pairs of noun-phrase entity mentions (cf. Section 5.4).

- **Rule mining:** NELL can learn a restricted form of Horn rules as soft constraints, such as: if two people are married they (usually) live in the same city. These rules are used to predict new statements and to constrain noisy candidate statements (cf. Section 8.3),

These learning and inference tasks are coupled: results of one task serve as training samples, counter-examples or (soft) constraints for other tasks. For example, one coupling constraint is that in an ensemble of type classifiers for noun phrases, all classifiers should agree on the predicted label. Another constraint is that instances of a class should also be instances of the class's superclasses, that instances cannot belong to mutually exclusive classes, and that domain and range constraints must be satisfied (cf. Section 8.3). In the same spirit, classifiers receive positive feedback if their classification corresponds to the predictions of NELL's learned rules. Overall, NELL has more than 4000 (instantiations of) learning tasks and more than a million (instantiations of) constraints.

NELL runs an infinite loop of two alternating steps, which are loosely modeled after the EM (Expectation Maximization) algorithm. In the E-step, NELL re-estimates the probability of each statement in its KB (called "beliefs"), by combining and reconciling the inputs from the different learners. In the M-like step, this confidence-refined KB is used to re-train the learners. To avoid semantic drift, humans intervene from time to time, and correct aberrant patterns or statements. Overall, NELL has learned more than 100 million confidence-weighted statements, including ca. 3 million with high confidence. Interestingly, in a sequence of such EM-style epochs, NELL can "unlearn" statements and rules that were accepted in previous rounds (i.e., lower its confidence in these). Nevertheless, subsequent epochs may learn these again (i.e., increase confidence).

As for its schema, NELL has also some means for discovering new relations, to be added to the ontology [397]. This is based on the *Path Ranking Algorithm* by [309, 308], which computes frequent edge-label sequences on paths of the KB graph for view discovery and rule learning (see Sections 7.3.3 and 8.3.3). The head of a newly learned rule can be interpreted as a new predicate. For example, NELL could potentially discover the relation `coveredBy` between musicians from path labels `created` (between musicians and songs) and $\texttt{performed}^{-1}$ (between songs and musicians).

The NELL website allows visitors to give feedback about statements, which not only corrects errors but also provides cues for future learning rounds. NELL could even proactively solicit feedback on uncertain statements via Twitter.

Last but not least, the NELL project also considered a limited form of *introspection*: automatically self-reflecting on the weak spots in the KB. By its ensemble learners, the confidence of statements and rules can be estimated in a calibrated way. This way, the learning machinery can be steered towards obtaining more evidence or counter-evidence on low-confidence beliefs and weak rules.

9.3.2 Lessons and Challenges

The overview article [395] provides specific references for all these methods, the learners, and the potential extensions. It also discusses lessons learned. The overarching insight is that NELL's principle of coupling different learners allows it to achieve good results with a small number of training samples for bootstrapping. Constraints are the key to tame noise and arrive at high-confidence statements.

NELL is also facing a number of open challenges:

- **Beware of the long tail:** Prominent instances of classes and relations are learned fast, but less frequently mentioned instances are more difficult to extract. This leads to the problem of when to cut off the extractions, and more generally, how to cope with the inevitable trade-off between precision and recall. The NELL KB is essentially a probability distribution over statements and rules with a long tail of low-confidence beliefs. Interpreting this *uncertain KB* is a challenge for downstream applications like querying and reasoning.

- **Learning convergence:** Another challenge is caused by NELL's never-ending learning, which makes it hard to tell when some task is completed (cf. Section 8.1). For example, the world has only ca. 200 countries, but NELL will continue trying to find more, learning spurious statements and rules, unlearning them later, and so on.

- **Degree of plasticity:** A key hypothesis has been that making most components of the system learnable ("plastic", as opposed to hard-coded) is beneficial for the output coverage and quality. Therefore, NELL is designed to learn its coupling constraints (as opposed to relying more on hand-crafted specification). However, it is not able to learn new ways of discovering new types from noun phrases and other cues. So it still lacks the ability to automatically expand its schema (cf. Chapter 7). The view discovery from path labels is an exception, but is limited to observing patterns over the pre-defined existing relations (see Section 7.3.3).

- **Entity canonicalization:** Although NELL can learn to identify synonymous entity names, it it suffers from insufficient support for entity linking (EL, see Chapter 5) to ensure unique representation of entities. For example, its learner for entity detection and typing yields noun phrases labeled "Elvis Presley", "Elvis Aaron Presley", "legendary Elvis Presley", "Elvis the King", "Elvis Presley 1935-1977", "Elvis lives", "Elvis Joseph Presley" and many more, without understanding that all these (except for the last one) are just different names for the same entity. This lack of canonicalization is an obstacle for some downstream applications (e.g., entity-centric analytics) and also makes it hard to maintain the KB in a consistent manner. For example, some of the seemingly different Elvis's above have died in Memphis, others have died in Nashville, and others are still alive (and hide on Mars).

9.4 Knowledge Sharing Community: Wikidata

Wikidata (https://www.wikidata.org) is a collaborative community to build and maintain a large-scale encyclopedic KB [600, 366]. It is currently the most comprehensive endeavor on publicy accessible knowledge bases.

Wikidata operates under the auspices of the Wikimedia foundation, and has close ties with other Wikimedia projects like Wikipedia, Wiktionary, and Wikivoyage. As of August 2020, the KB contained 88 Million entities, and has 23,000 active contributors.

For this survey article, three aspects of Wikidata are especially relevant: i) the way how knowledge in Wikidata is organized, ii) the way how the schema evolves in this collaborative setting, and iii) the role that Wikidata plays as a hub for entity identification and interlinkage between datasets.

9.4.1 Principles and History

Wikidata was launched in 2012, following earlier experiments with collaborative data spaces such as Semantic MediaWiki [296]. The motivation for collecting and organizing structured data in the Wikimedia ecosystem was twofold: (i) centralizing inter-language linking and (ii) centralizing infobox data. Wikidata should be used for automatically generating these across all Wikipedia language editions, thus simplifying maintenance and ensuring consistency. As of August 2020, the first goal has been reached, and the second one is getting closer.

The population of the Wikidata KB started out with human contributors manually entering statements. In 2014, Google offered the content of its, then phased out, Freebase KB [51] for possible import into Wikidata: 3 billion statements about 50 million entities. The Wikidata community refrained from automated import for quality assurance; instead a tool was created that allowed editors to validate (or discard) individual statements before insertion into the KB [581]. As a result, ca. 17 million statements about 4.5 million entities were added to Wikidata. This small fraction of the enormous volume of Freebase underlines the very high quality standards that the Wikidata community exercises (including the requirement for reliable references to support statements). The rapid growth of Wikidata in subsequent years has been largely based on its human contributors, but also used bulk imports with humans-in-the-loop for quality assurance (see below).

Beyond this Wikipedia-centric usage, it turned out that Wikidata became useful for many other purposes as well: interlinking and enriching data from public libraries and archives, and managing biomedical and scholarly data (see Section 9.4.3).

9.4.2 Data Model and KB Life-Cycle

Entities (aka. Items) and Statements (aka. Claims):
Wikidata's data model [600] revolves around entities, SPO triples, and qualifiers. *Entities*, called *items* in Wikidata jargon, have language-independent Q-code identifiers (e.g., Q392 for *Bob Dylan*, or Q214430 for *Like a Rolling Stone*). Figure 9.1 shows an excerpt of the Wikidata page for Bob Dylan, annotating some of the key concepts of the data model.

SPO triples, called *claims*, are statements about entities. Following the RDF model, subjects are entities with Q-codes and objects can be entities or literals. The predicates come from a manually specified set of about 7000 properties, identified via language-agnostic P-codes (e.g. P569 for date of birth, P26 for spouse, or P577 for publication date). This predicate space and the entities alike are continuously and collaboratively expanded by the contributors in the Wikidata community.

Wikidata circumvents the restrictions of triple-based knowledge representation by reification (e.g., Macron's inauguration as French president is itself an entity) and by *qualifiers* for refining SPO triples (cf. Section 2.1.3). Qualifiers are a predicates that enrich triples with context, about sources, dates, reasons, etc. For example, the `spouse` property comes with qualifiers for wedding date and divorce date. Awards, such as Bob Dylan winning the Nobel Prize, have qualifiers for date, location, field, prize money, laudation speaker, etc.

Taxonomy and Consistency:
Wikidata items are organized into classes by use of the *instanceOf* property, which in turn are organized using *instanceOf* and *subClassOf*. There is no hard distinction, though, between instances and classes; for example, `Buddha` (https://www.wikidata.org/wiki/Q7055) is both an instance of `religious concept` and a subclass of `religious ecstasy` and `person` (as of August 27, 2020), and the latter leads to super-classes like subject, agent, item, individual. The complexity, difficult interpretability and potential inconsistency of the Wikidata class system was specifically addressed by YAGO 4 (see Section 9.1).

An important principle of the collaborative community is to allow different perspectives. Therefore, Wikidata refers to statements as *claims*. These do not necessarily capture a single view of the world, with universally agreed-upon facts. Rather it is possible (and

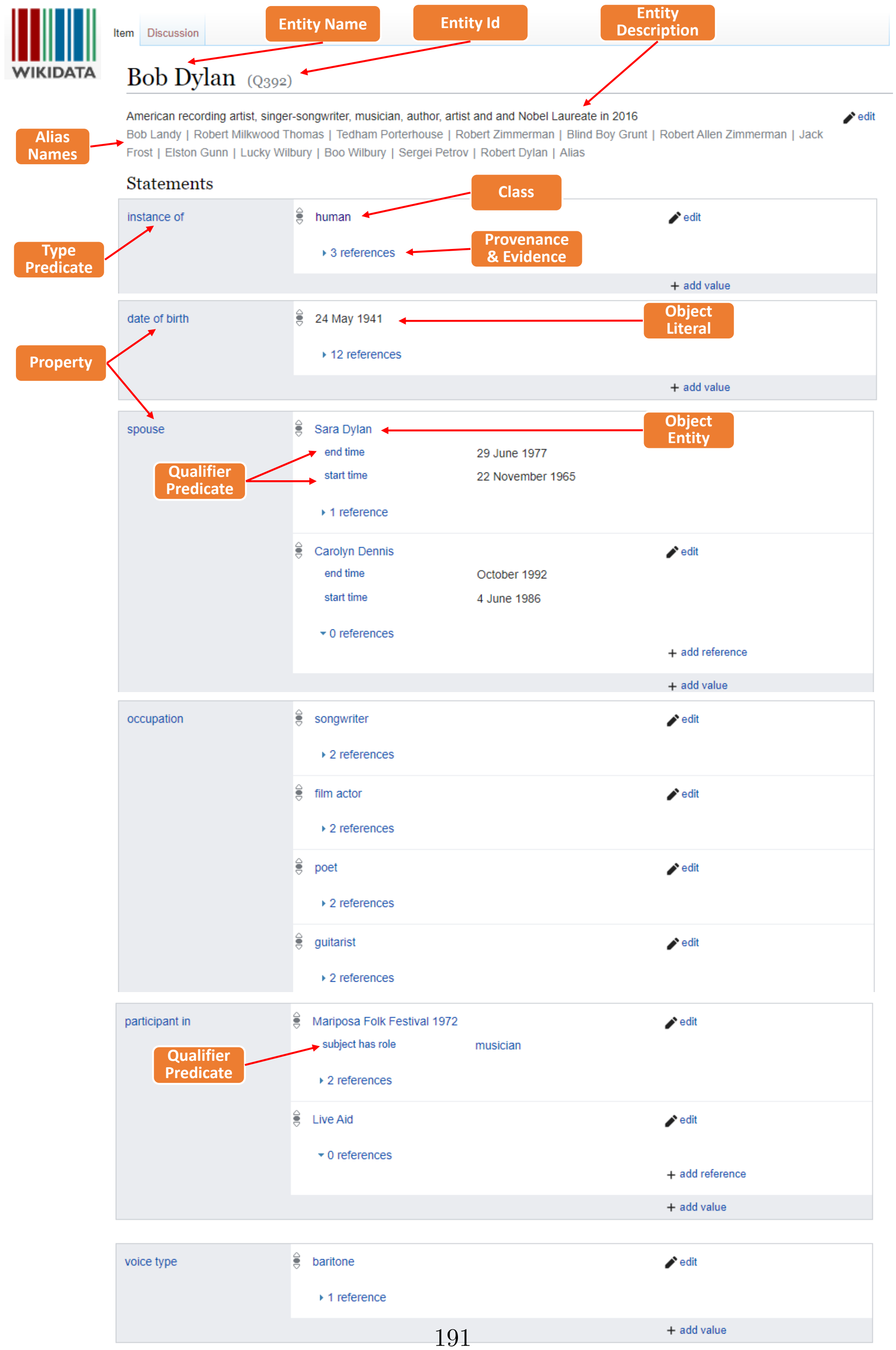


Figure 9.1: Excerpt of Wikidata page for *Bob Dylan* (www.wikidata.org/wiki/Q392, August 31, 2020)

often happens) that alternative standpoints are inserted as separate statements for the same entity and property. For example, it is accepted that Jesus has more than one birth date, and the same holds for the death date of the mountaineer George Mallory (who died somewhere on Mount Everest with unknown precise date). In principle, Wikidata would even tolerate entering alternative death places for Elvis Presley (including perhaps Mare Elysium on Mars), but the community does have moderators who may intervene in such a case.

Wikidata also supports a rich portfolio of *consistency constraints*, including type constraints for properties, functional dependencies (called single-value constraints) and more. However, by the philosophy of capturing potentially conflicting claims, the constraints are not rigorously enforced. Instead, they serve to generate warnings when entering new data; so it is up to the contributors to respect the constraints or not. Also, users can systematically look at constraint violations and make corrections as deemed appropriate.

KB Life-Cycle:
Like Wikipedia, Wikidata follows a collaborative process, under which both data and schema continuously evolve. For inserting entities and statements, an important criterion is to include references to reliable sources of evidence. For Bob Dylan (as of August 2020), for example, these include mostly digital libraries, archives and LOD sources such as BBC (http://www.bbc.co.uk/music/sevenages/artists/bob-dylan/), Musicbrainz (https://musicbrainz.org/artist/72c536dc. . .) and also Wikipedia articles. For some statements beyond the standard biography and creative works, also news articles and authoritative websites such as https://www.biography.com/people/bob-dylan-9283052 are cited as references. An example is Dylan's unmarried partnership with Joan Baez, supported by citing a new article from https://www.dailymail.co.uk/. Schema edits, on the other hand, are typically subject of community discussions. Frequent issues are, for instance, whether a proposed property type will be re-used often enough, whether a a property could also be expressed by class membership, and what kinds of constraints should be imposed.

Regardless of its initial intention to support Wikipedia, Wikidata now holds statements about many entities that are not present in Wikipedia at all. This is largely data imported from so-called GLAMs (an acronym for galleries, libraries, archives, museums), and government organizations that provide Open Data. Typically, this involves adding an entity, like a book author or public school, and a set of statements for their identifiers in the original data sources. The data imports go through a semi-automated process: first discussed and approved by the community, then carried out by bots. Typical issues for approval are whether the data is of sufficient interest for a KB at all, how data quality is assured, and how to interlink the new data with the existing content of the KB. Import procedures are supported by a variety of tools, such as OpenRefine (https://openrefine.org) and Mix'n'match (https://mix-n-match.toolforge.org).

9.4.3 Applications

Beyond Wikipedia, Wikidata is utilized in a range of applications.

Galleries, Libraries, Archives and Museums (GLAMs):
These stakeholders are major proponents of open and interlinked data, Their entities, like authors, artists and their works, are increasingly captured in Wikidata, typically with identifiers that point to the original repositories. For some entities, these identifiers constitute the majority of the statements in Wikidata. GLAMs have high interest in such interlinking, opening up their contents for collaborative enrichment and better visibility in search engines. An ideal enrichment would follow the "round-trip" pattern: Wikidata imports entity identifiers, say about a lesser known painter from a museum, the Wikidata community augments this with facts about the painter's biography, and this added value can be easily combined with the museum's online contents about the painter's works [121].

Scholarly Knowledge:
Knowledge about scientific publications, authors and their organizations is another use case that is gaining importance. Notable projects include CiteSeerX [628, 11], SemanticScholar [12, 353], AMiner [578, 607], Open Research Knowledge Graph [430, 57], and Scholia [426]. The last one, Scholia, is directly based on Wikidata. All these endeavors aim to provide tools for the semantic analysis of scholarly topics, author networks, bibliometric measures of impact and more, as open alternatives and value-added extensions to prevalent services like Google Scholar, Microsoft Academic or publisher services [538, 216, 156].

Entity Identification:
Another desirable purpose of KBs is to provide master data for *entity identification* and *cross-linkage* between resources. Wikidata is taking up a central role in the Web of Linked Open Data (LOD, see Section 9.6). Wikidata identifiers are becoming widespread in interlinking datasets and knowledge repositories. In turn, a large fraction of Wikidata statements are about external identifiers, such as *TwitterID, VIAF-ID* or *GoogleScholarID.*

Life Science Knowledge:
The life sciences – biomedicine and health – is a specific domain where Wikidata has the potential to become a data hub [605]. On one hand, the Wikidata KB contains a large amount of statements about diseases, drugs, proteins etc. On the other hand, its rich coverage of identifiers as links to other repositories (see above) supports combining from different biomedical sources in a user-friendly manner (e.g., for data scientists in the health area).

9.4.4 Challenges

Wikidata is today's most prominent endeavor on collaborative knowledge engineering. Still, it faces a number of challenges in its future advances.

Quality Assurance and Countering Vandalism:
These are never-ending concerns for collaborative projects. A good strategy requires balancing openness of contributions and moderated approvals. Allowing edits by any contributor has the risk of introducing errors but also the advantage that errors can be quickly caught and corrected by others. Staggered review and approval, on the other hand, can prevent blatant errors, at the expense of slowing down the KB growth, though. In addition to vandalism, concerns are also raised over more subtle content distortions with commercial or political interests.

Evolving Scope and Focus:
The scope and focus of Wikidata are repeatedly coming into discussion, especially when new data imports are discussed. In 2019, for instance, debates revolved around the importing of scholarly data (mostly identifiers for authors and publications), which currently makes up 40% of Wikidata's entities. Such imbalances could possibly bias functionality for search and ranking, and puts strain on Wikidata's infrastructure, potentially at the cost of other stakeholders. Therefore, decisions on in-scope and out-of-scope topics are a recurring concern.

Schema Stability:
Wikidata's collaborative processes and continuous evolution are better able to keep up with an evolving reality than any expert-level modeling of classes and properties. However, this implies that applications relying on Wikidata require continuous monitoring, as changes in property definitions and taxonomic structures can break queries. Finding the right balance between the community's grassroots contributions and controlling the quality and stability of the KB schema will remain a challenge.

Data Duplication:
Redundancy, and the resulting risk of inconsistency, are further issues that arise from the principle of a collaborative community. At the entity level, this is well under control by the self-organization among editors which captures duplicates quickly and resolves them. However, redundancy is an issue for types and properties. By independent edits, properties can be duplicated in forward and backward direction, such as *parent / child*, *has part / is part of*, and *award received / recipient*. Similarly, existential information is stored in separate properties, such as *child / number of children* and *episode / number of episodes*. These make querying more complex, and most critically, may easily cause inconsistencies.

9.5 Industrial Knowledge Graphs

Knowledge graphs, or *KGs* for short, is the industry jargon for knowledge bases – a widely used but oversimplifying term as KBs comprise much more than just binary relations. KGs started taking an important role in industry in 2012, the year when Google launched knowledge-based search under the slogan "things, not strings" [536]. The Google KG started from Freebase [51], a KB built by Metaweb, acquired by Google in 2010. In the same year, Amazon acquired Evi (formerly True Knowledge) [587], whose KG laid the foundation for Alexa question answering. Knowledge graphs are broadly used in search engines like Google, Bing and Baidu, in question answering such as Apple Siri, Amazon Alexa and Google Assistant, and in e-Commerce at Alibaba Taobao, Amazon, eBay, Walmart and others. Another early player in industrial KGs has been Wolfram Alpha [248], which provided services to Apple, Amazon, Microsoft, Samsung and others. Building an authoritative knowledge graph with comprehensive and high-quality data to support the broad range of applications has been a hot topic for industry practice in the past decade [429].

The biggest success for knowledge bases in industry is mainly for *popular domains* such as music, movies, books and sports. Efforts to collect *long-tail knowledge* started much later (around 2015), but have already been successful in facilitating users to find answers for their hobbies such as yoga or cocktails. Gathering and organizing *retail product knowledge* started quite recently (2017), facing many obstacles but already bearing fruits. We next describe the efforts and progress for each of these three aspects.

9.5.1 Curating Knowledge for Popular Domains

The first pot of gold for knowledge collection in industry is Wikipedia. As discussed in Chapter 3, Wikipedia is a premium source and great starting point with its huge number of entities, informative descriptions and rich semi-structured contents. Wikipedia data has played an important role in Google KG, Bing Satori, Amazon Evi, and presumably more.

The knowledge graphs are then extended on a set of popular domains: large domains include *Music, Movies, Books, Sports, Geo, etc.*, and medium domains include *Organizations, People, Natural World, Cars, etc.* Two common features for such domains make them pioneer domains for knowledge collection.

- First, there are already rich data sources in (semi-)structured form and of high quality. For example, IMDB (https://www.imdb.com/interfaces/) is a well-known authoritative data source for movies, and MusicBrainz (https://musicbrainz.org/doc/MusicBrainz_Database) is an authoritative source for music with open data for free download. Big companies also license data from major data providers, and often have their own data sets (for example, Google has rich data on books, locations, etc.).
- Second, the complexity of the domain schema is manageable. Continuing with the movie

domain, the Freebase knowledge graph contained 52 entity types and 155 properties for this domain. The schema (aka. ontology) to describe these types and properties can be manually specified by a knowledge engineer within weeks, especially by leveraging existing data sources.

A big challenge in this process is to integrate entities and properties from Wikipedia with domain-specific data sources. This requires alignment between schemas/ontologies (see Section 3.3) and entity matching (see Section 5.2). Schema alignment is carefully curated manually; this is affordable because the size of the schema/ontology for each domain is manageable. On the other hand, manually linking entities that refer to the same real-world person or movie is unrealistic, at the scale of many millions of entities. This calls for automatic entity linking (EL) as discussed in Chapter 5 (see especially Section 5.2). To meet the high bar for KB accuracy, the entity linkage needs to have very high precision, obtained by sacrificing recall to an acceptable extent, and by manually checking cases where the EL method has low confidence.

9.5.2 Collecting Long-Tail Knowledge

Long-tail domains are those where the entities are not globally popular; thus oftentimes the number of entities is small: thousands or even hundreds only. Examples of tail domains include gym exercises, yoga poses, cheese varieties, and cocktails. Although a tail domain may not be popular by itself, given the large number of tail domains, they can be collectively impactful, for example, addressing people's hobbies – an important part of life.

Entity distribution among head, torso, and tail domains observes the power law; that is, a few head domains cover a huge number of entities, whereas a huge number of long-tail domains each covers a fairly small number of entities. The huge number of long-tail domains and the diversity of their attributes make it impossible to collect knowledge from a few domain-specific premium sources as we do for head domains. Likewise, there is no source that covers a significant fraction of different long-tail domains. An empirical study with manual curation of four long-tail domains (cheese varities, tomatoes, gym exercises, yoga poses) showed that many entities of interest can be found on websites, but these entities neither exist in Wikipedia nor in the Freebase KB [327].

Millions of tail domains and no single data source to cover all domains call for a different solution for collecting long-tail knowledge. We next describe two different approaches: tooling for curation (Section 9.5.2.1) and automatic extraction from a huge variety of websites (Sections 9.5.2.2 and 9.5.2.3).

9.5.2.1 Tools for Long-tail Knowledge Curation

One effective method to extract long-tail knowledge is to provide tools that support knowledge engineers on the manual curation of long-tail domain knowledge. The Google

Knowledge Graph and projects at Amazon that feed into Alexa took this approach.

The process comprises five steps:

1. Identify a few data sources for each domain.
2. Define the schema for the domain according to data in these sources.
3. Extract instance-level data from the sources through annotation tools and hand-crafted patterns.
4. Link the extracted knowledge to existing entities and types in the knowledge base.
5. Insert all new entities and acquired statements into the knowledge base.

Each step involves manual work and human curation. It can easily take a knowledge engineer a few weeks to curate one long-tail domain.

There are studies on how to accelerate each step in this process and reduce the manual work. The work of [613] proposed the *MIDAS* method for discovering web sources of particular value for a given domain. The core insight from MIDAS is that even if automatic extraction methods may not yield sufficiently high quality overall, they can provide clues about which websites contain a large amount of relevant contents, allow for easy annotation, and are worthwhile for extraction (see the discussion on source discovery in Section 6.2.2.5).

A challenge in harvesting these sources, however, is that a website often includes facts about multiple groups of entities, having different schemas and requiring different extraction templates (see Section 6.3). MIDAS discovers groups that contain a sufficient number of entities and statements that are absent from the KB one wishes to augment, such that the extraction benefits outweigh the efforts. MIDAS also helps to identify interesting domains, such as medicinal chemicals or US history events.

Another study [327] aimed to identify extraction errors from particular sources by verifying the correctness of the collected knowledge. This is based on an end-to-end knowledge verification framework, called *FACTY*. The method leverages both search-based and extraction-based techniques to find supporting evidence for each triple, and subsequently predicts the correctness of each triple based on the evidence through *knowledge fusion* [129] (see Sections 8.1.3 and 8.5.1). Various types of evidence are considered, including existing knowledge bases, extraction results from websites, search-engine query logs, and results of searching subject and object of SPO triples. In the study of [327], this technique achieved high recall: positive evidence was found for 60% of the correct triples on 96% of the entities in the examined four tail domains.

Unfortunately, this method also finds positive evidence for many incorrect statements; so the outlined approach suffered from low precision (fraction of verified triples that are truly correct). To distinguish correct and wrong triples, knowledge fusion estimates the correctness of each triple, taking into consideration the quality of the evidence sources and the reliability of the techniques for obtaining evidence. With this additional quality assurance, the percentage of correct statements that could be verified dropped from 60%

to 50%, but the precision increased to 84% (i.e., among all triples that are verified to be correct, 84% are indeed correct) [327]. This is an enormous support and productivity boost for human curators.

9.5.2.2 Google Knowledge Vault

Tooling can speed up the collection of long-tail knowledge; however, the solution still does not scale to hundred thousands of domains. For high coverage of many domains, it is inevitable to apply information extraction (IE) methods to millions of diverse websites. Among such large-scale endeavors, *Google Knowledge Vault (KV)* is a prominent one [129, 130].

Knowledge Vault differs from other projects in two important aspects:

- First, KV extracts knowledge from four types of web sources:
  - o text documents such as news articles,
  - o semi-structured DOM trees, where attribute-value pairs are presented with rich visual and layout features,
  - o HTML tables with tabular data embedded in web pages, and
  - o human-annotated pages with microformats according to ontologies like *schema.org* [193].
- Second, KV addresses the low quality of the extraction by applying knowledge fusion techniques (see Sections 8.1.3 and 8.5.1).

KV applied 16 types of extractors on these four types of web contents. For text and DOM trees, extraction techniques were used as discussed in Sections 6.2 and 6.3. In particular, KV first identified Freebase entities in a sentence or semi-structured page, and then predicted the relation between a pair of entities. The training data was obtained from Freebase, using distant supervision. To this end, KV utilized the *Local Completeness Assumption (LCA)* (see Section 8.2): when a subject-property combination has at least one object in the KB, then the set of triples with this SP combination is complete. Conversely, any choice of object O for the same SP pair such that SPO is not in the KB, must be false. This has been key to generating also negative training samples, and turned out to be decisive for extraction quality [129].

For web tables, schema mapping and entity linking were employed to map rows and columns to entities and properties, respectively (cf. Section 5.7). For pages with microformat annotations, KV extracted statements that follows the *schema.org* ontology and then transformed this to the Freebase schema. These mapping were manually specified, as they involved only a few tens of property types.

Experimental Findings:
The above methods extracted 2.8 billion SPO triples from over 2 billion web pages. Among the four source types, DOM trees for HTML-encoded web pages contributed 75% of extracted statements. This is not surprising because each semi-structured website normally has one or more back-end databases that yield many instances. In comparison, web tables contributed the smallest number of extractions, less than 5%. This is partly because web tables are hand-crafted and thus contain only few rows, in contrast to tables or lists that are generated from back-end databases. Another reason is that the extraction techniques faced limitations on aligning column headers with KB properties (as web tables often have generic and highly ambiguous headers such as "Name" or "Count").

Despite the large volume of extractions, extraction quality initially was very low, with precision around 10%. KV solved the problem by applying *knowledge fusion* techniques in an additional curation phase (see Chapter 8, especially Section 8.5.1). First, it trained a logistic regression model to accept or reject a triple based on from how many web pages it was extracted and by how many extraction models it was returned. Second, KV employed the *Path Ranking Algorithm* [310] for deduction (see Section 8.3.3), to assess the plausibility of a triple according to paths between the S and O entities. Third, KV used a neural network for link prediction, based on KG embeddings (see Section 8.4). Note that all three techniques were used conservatively here, to prune false positives (not for inferring additional triples).

Among the three approaches, each of them achieved over 90% area-under-the-ROC-curve; all three together reached nearly 95%. In the end, among a pool of 1.6 billion candidate triples, 324 million (20%) were assessed to have confidence 0.7 or higher, and 271 million (17%) had confidence 0.9 or higher – high-quality candidates to be considered for KB augmentation.

Lessons Learned:
Despite the promising results, KV did not significantly contribute to the Google Knowledge Graph, mainly because the number of new triples meeting the very high bar for precision was not sufficiently high (the Google KG expected 99% precision). There are several reasons. First, the extractions are restricted to existing entities and properties. Since the Google KG already has a high coverage of top entities and their properties, missing relations between popular entities were not many. Second, the first-stage extraction quality was too low. The knowledge-fusion stage could prune false positives, but could not add any correct extractions. Third, evaluation in the wild may not be as good as estimated from the local completeness assumption (LCA, see above). It is common to observe different quality when the training data and test data have different distributions, and that is more likely to happen when the training data is not randomly sampled and makes certain assumptions.

Nevertheless, the KV data found an interesting application for estimating the quality of web sources by a *Knowledge-Based Trust* model [131] (see Section 8.5.1). The underlying intuition is that the trustworthiness of a web source can be measured by the accuracy of the triples provided by the source, which can be approximated from the confidence-weighted number of triples extracted from the source. A probabilistic graphical model was devised to compute the probability of a triple being correct, the probability of an extraction being correct, the precision and recall of an extractor, and the trustworthiness of a data source.

The method computed trustworthiness scores for 5.6 million websites and led to interesting observations. First, most of the sources have a score above 0.5, and some reach 0.8. Second, the trustworthiness scores are orthogonal to PageRank scores, which is based on the popularity of a website rather than its content quality. The knowledge-based trust method is the basis for fact verification in knowledge curation tools, described in Section 9.5.2.1.

9.5.2.3 Amazon Ceres

Amazon Ceres [354, 355, 356] is another large-scale effort to collect long-tail knowledge by web extraction. Ceres made two design choices that deviate from the rationale of Knowledge Vault.

- First, instead of relying on knowledge fusion to clean up extracted statements, it focuses on improving extraction quality from the beginning. Ceres improved extraction precision from semi-structured sources from 43% in KV to over 90%, a quality reasonable for industrial-strength services.
- Second, Ceres also extracts knowledge about new entities, and even new properties. This way, it extracts head knowledge for tail entities, instead of tail knowledge for head entities, and thus has more room to satisfy users' long-tail knowledge needs.

These differences make Ceres suitable for industry production, used at Amazon to collect long-tail knowledge for Alexa and product knowledge for retail.

Ceres focuses on only one type of web content: semi-structured web pages. Recall that DOM-tree-based contents contributed 75% of extractions and 94% of high-confidence extractions in Knowledge Vault; so is a natural top choice among the different types of web contents. The dense high-quality information in semi-structured sources makes it easier for extraction and entity linking, supporting the two design choices above.

On the other hand, semi-structured contents pose the challenge that every website follows a different template (or a set of templates). So the model trained on one website normally does not carry over to another website. Ceres devised a suite of extraction techniques to address this challenge: Ceres [354] for properties specified in the KB schema (referred to as "Closed IE"), OpenCeres [355] for Open IE (see Chapter 7), and ZeroShotCeres [356] for extraction from unseen websites and even unseen subject domains. We next outline the three systems.

Ceres for Schema-based "Closed IE":
Ceres extracts knowledge according to an existing schema/ontology. Training a different model for a different website would require a large amount of training data, posing a bottleneck. Ceres solves this problem by generating the training data automatically using existing knowledge as seeds. This is the distant supervision technique described in Section 6.3.1 and widely used in large-scale knowledge harvesting. However, compared to Google Knowledge Vault, for example, Ceres is able to double the extraction precision with two significant enhancements:

- First, Ceres focuses on entity detail pages (see Section 6.3.1), each of which describes an individual entity. It employs a two-step annotation algorithm that first identifies the entity that is the primary subject of a page, and then annotates entities on the page that have relations with that entity in the seed knowledge. This prevents a large number of spurious outputs.
- Second, Ceres leverages the common structure among property-object pairs within a web page, and across pages from the same website, to further improve the labeling quality.

With improved labeling from distant supervision, and thus better training data, Ceres is able to extract knowledge from DOM trees in websites about long-tail domains with precision above 90% [354].

Open-IE-based Ceres:
OpenCeres [355] and *ZeroShotCeres* [356] address the Open IE problem, with the goal of extracting properties that do not exist in the schema/ontology (see Chapter 7). These systems are among the first OpenIE approaches specifically geared for semi-structured data, and did so by exploring both structural and layout signals in web pages. Between them, OpenCeres trains different models for different websites, and requires the website to contain seeds from the existing KB. ZeroShotCeres, as the name indicates, can extract statements from a new website in a completely new domain, by training a universal model that applies to all websites.

OpenCeres explores similarities between different property-object pairs within a page. The idea is that the format for an unknown property (e.g., `coveredBy` for songs) is often similar to that for a known one (e.g., `writtenBy`). It applies semi-supervised label propagation to generate training data for unseen properties (see Section 7.3.2).

ZeroShotCeres [356] extends this approach by considering similarities between different websites. The underlying intuition is that despite different websites having different templates, there are underlying commonalities in font, style, and layout. ZeroShotCeres trains a *Graph Neural Network* to encode patterns common across different websites. Thus, the learned model can generalize to previously unseen templates and even new vertical domains.

9.5.3 Collection of Retail Product Knowledge

Recent years have been witnessing KB applications in a new domain: the retail product domain. Similar to generic domains, product knowledge can improve product search, recommendation, and voice shopping. However, this new domain presents specific challenges.

Challenges:
First, except for a few categories such as electronics, structured data is sparse and noisy across nearly all data sources. This is because the majority of product data resides in catalogs from e-Business websites such as Alibaba, Amazon, Ebay, Walmart, etc., and these big players often rely on data contributed by retailers. In contrast to stakeholders for digital products like movies and books, in the retail business, contributing manufacturers and merchants mainly list product features in titles and textual descriptions instead of providing structured data [666, 634]. As a result, structured knowledge needs to be mined from textual contents like titles and descriptions. Thousands of product attributes, billions of existing products, and millions of new products emerging on a daily basis require fully automatic and efficient knowledge discovery and update mechanisms.

Second, the domain is complex in various ways. The number of product types is towards millions, and there are sophisticated relations between the types like subclasses (e.g., swimsuit vs. athletic swimwear), synonyms (e.g., swimsuit vs. bathing suit), and overlapping types (e.g., fashion swimwear vs. two-piece swimwear). Product attributes vastly differ between types (e.g., compare TVs and dog food), and also evolve over time (e.g., older TVs did not have WiFi connectivity). All of these make it hard to design a comprehensive schema or ontology and keep it up-to-date, thus calling for automatic solutions. Open catalogs like https://icecat.biz/ do not provide extensive schemas, and face the same issue that fine-grained types and properties are merely mentioned in textual documentation.

Third, the huge variety of different product types makes it even harder to train models for knowledge acquisition and curation. Product attributes, value vocabularies, text patterns in product titles and descriptions widely differ for different types. Even highly related product types can have rather different attributes. For example, *Coffee* and *Tea*, which share the same parent *Drink*, describe packaging sizes by different vocabularies and patterns, such as:

"Gound Coffee, 20 Ounce Bag, Rainforest Alliance Certified" vs.

"Classic Tea Variety Box, 48 Count (Pack of 1)").

Training a single one-size-fits-all model for this huge variety is hopeless. On the other hand, collecting training data for each of the thousands to millions of product types is prohibitively expensive.

Research on product knowledge graphs is pursued by major stakeholders including Alibaba Taobao, Amazon, eBay and Walmart [429, 360, 133, 633, 450]. Next, we exemplify current approaches to these challenges by discussing methods for building and curating the Amazon Product Graph.

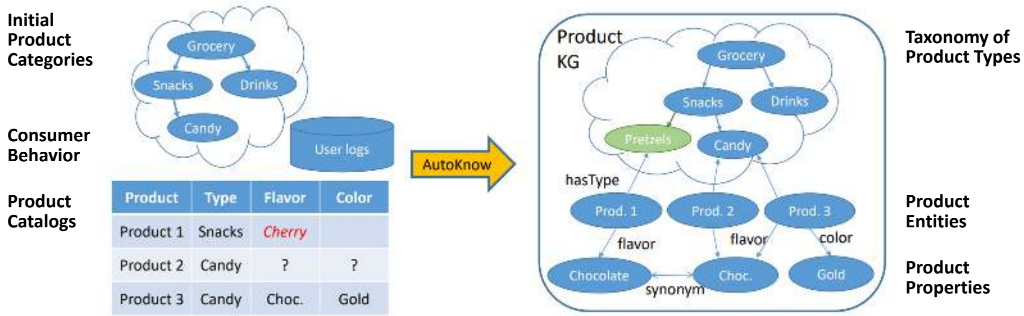


Figure 9.2: Input and Output of the AutoKnow System

9.5.3.1 Amazon Product Graph

Amazon is building a Product Graph for retail products. Figure 9.2 illustrates the inputs and outputs for this endeavor. The inputs include pre-existing categories, catalogs and signals from consumers' shopping behaviors, like queries, clicks, purchases, likes and ratings. The output is a KB that comprises an enriched and clean taxonomy and canonicalized entities with informative properties. To distinguish the product knowledge graph from a traditional KB (or KG), the output is referred to as a "broad graph". This is a bipartite graph, where one side contains nodes representing products and the other side contains nodes representing product properties, such as brands, flavors, and ingredients (lower part of the right side of Figure 9.2).

Amazon's knowledge collection system for its product KG is called *AutoKnow* [133]. The system starts by building a product type *taxonomy* and deciding on applicable product properties for each type. Subsequently, it infers structured tripes, cleans up noisy values, and identifies synonyms between attribute values (e.g., "chocolate" and "choc." for the `flavor` property).

AutoKnow is designed to operate fully automatic at scale, suited for a huge number of entities and types in the retail domain. For training its machine-learning models, it leverages existing catalogs and customer behavior logs. These are used to generate training samples, eliminating the need for manual labeling. The learned models can generalize to new product types that have never been seen during training (see [133] for technical details).

A few techniques play a key role for scalability and generality:

- AutoKnow devised a *Graph Neural Network* for machine learning, which lends itself to the structure of the product KG.
- It takes initial product categories as input signal to train models for building the KG taxonomy.
- It heavily relies on distant supervision and semi-supervised learning, to minimize the burden of manual labeling.

- It jointly infers facts about properties and synonymous expressions for types, properties and values, to cater for the widely varying vocabularies by consumers.

9.5.4 Outlook

KB technology has been highly impactful in Internet-centric industry, like search engines, online shopping and social networks [429]. As of 2020, it remains a hot topic in industry as the bars for quality and coverage are raised and use cases are being expanded (see, e.g., [528, 245]).

The technology has largely been driven by applications in mainstream Internet business, but there are many use cases also for other domains such as health, materials, industrial plants, financial services, and fashion (e.g., [501, 556, 251, 466, 277]). Moreover, as scalable data platforms and machine learning are becoming commodities and affordable to small players as well, KB technology is also advanced by startups and specialized companies (e.g., [656, 380]).

9.6 Ecosystem of Interlinked Data and Knowledge

The *Semantic Web* is the umbrella term for all semantic data and services that are openly accessible via Web standards, including several knowledge bases. The design principles that KBs satisfy to be compliant with Semantic Web standards are the following:

- **Uniform machine-readable data representation:** This principle is implemented by using the RDF/RDFS standards for representing knowledge as statements in triple form (see Section 2.1).

- **Self-descriptive data:** For semantic interpretation, there is no distinction between the instance-level data and the schema. The KB can can describe its schema, constraints, reasoning rules, and taxonomy/ontology in a single model. The relevant standards are RDFS, SHACL, and OWL (see Section 8.3).

- **Interlinked data:** A KB can link to other KBs via statements that reference entities in other KBs, or directly at the entity level, by the `owl:sameAs` property (cf. Section 5.2), and can be accessed using standardized protocols.

This last point lifts the KB into an ecosystem of connected KBs and other datasets, which is called the Web of Data, Linked Data, or (if the data is openly available) Linked Open Data (LOD), or the LOD cloud [225, 244]. For this to work, the KBs have to provide entity identifiers that are globally unique – so that the Elvis Presley in DBpedia can be distinguished from the Elvis Presley in YAGO. For this purpose, each KB is identified by a *uniform resource identifier* (URI, a generalized form of URL), such as *https://yago-knowledge.org* for YAGO. Each entity in the KB is identified by a local name in the name space given by that URI, as in *http://yago-knowledge.org/resource/Elvis_Presley* for Elvis in YAGO. This requirement is already part of the RDF standard. To avoid cumbersome identifiers, RDF allows abbreviating URI prefixes. For example, by the abbrevation *yago:* for *http://yago-knowledge.org/resource/*, we can simply write *yago:Elvis_Presley*.

Interestingly, a KB can contain statements about entities from other KBs. For example, DBpedia can assert that its Elvis Presley belongs to the class *singer* that is defined in YAGO, by the statement

```
dbpedia:Elvis_Presley rdf:type yago:Singer
```

This has the advantage that a KB can re-use the vocabulary of another KB. The prefix “rdf:” refers to the standard vocabulary of RDF, which can be seen as another KB in the same unified framework. The same applies to the OWL and SHACL vocabularies. There are also vocabularies with extensive modeling of semantic classes, like schema.org [193] (used in Section 9.1.

The final ingredient for the Web of Data is making these URIs *dereferenceable*: When we access the URI of Elvis in DBpedia, the server of DBpedia replies with a piece of RDF data about Elvis. In this reply, the client can find the URI of a class in YAGO, which it can follow in the same way. By this mechanism, the machine can seamlessly “surf the Semantic Web” across all its knowledge and data bases, analogous to humans surfing the HTML-based Web.

The most important links between KBs are *owl:sameAs* links. They are used to assert that an identifier in one KB refers to the same real-world entity denoted by another identifier in another KB:

```
dbpedia:Elvis_Presley owl:sameAs yago:Elvis_Presley
```

These links enable applications to complement the data found about an entity in one KB with the data from another KB. As of 2019, more than 1200 data and knowledge resources are interlinked this way (see https://lod-cloud.net).

Another useful element of the Semantic Web are annotation languages like RDFa, Microdata, or JSON-LD for augmenting HTML documents with RDF data [381]. Search engines use such annotations to identify, for example, product reviews with their scores, shops with their locations, or movies with their showtimes. Such micro-data appears in billions of web pages, with increasing trend (see, e.g., http://webdatacommons.org/structureddata/). While a large portion of annotations concern website meta-data, the most widely used class of HTML micro-data is *schema:Product*.

Finally, the Semantic Web has pushed forward the adoption of shared vocabularies.

Perhaps the most prominent effect is *schema.org* [193], a vocabulary for thousands of classes, developed by Google, Microsoft, Yahoo and Yandex and standardized by the World Wide Web Consortium.

10 Wrap-Up

10.1 Take-Home Lessons

This article discussed concepts and methods for the task of automatically building comprehensive knowledge bases (KBs) of near-human quality. This entails two strategic objectives:

- a very high degree of *correctness*, with error rates below 5 percent or even under 1 percent, and
- a very large *coverage* of entities, their semantic types and their properties, within the scope and intended usage of the KB.

Low-hanging Fruit First: The two goals about correctness and coverage are in tension with each other. The first goal suggests conservative methods, tapping largely into premium sources and using high-precision algorithms such as rules and (learned) patterns. Premium sources (e.g., Wikipedia, IMDB, Goodreads, MayoClinic etc.) are characterized by i) having authoritative high-quality content, often in semi-structured form with categories, lists, tables, ii) high coverage of relevant entities, iii) clean and uniform style, structure and layout, and, therefore, iv) being extraction-friendly. The second goal calls for aggressive methods that can discover promising sources, identify long-tail entities and extract interesting properties beyond basic facts. Typically, this involves powerful but riskier methods such as Open IE and deep neural networks

Such aggressive methods can often leverage the outcome of the conservative stage by using high-quality data from an initial KB as *seeds for distant supervision.* This mitigates the typical bottleneck of not having enough training samples. The bottom line is that KB construction should consider harvesting "low-hanging fruit" first and then embark on more challenging methodology as needed.

Core Knowledge on Entities and Types: Entities and their semantic types form the backbone of every high-quality KB. Attributes and relations are great assets on top, but they can shine only if the entity-type foundation is proper. Search engines and recommender systems often need only entities, class memberships, and entity-entity relatedness. This is why this article first emphasized the construction of *taxonomic knowledge* and the *canonicalization of entities* (Chapters 3 through 5). Based on this sound foundation, populating the KB with attributes and relations (Chapters 6 and 7) yields great value for advanced use cases such as expert-level QA or entity-centric analytics over properties.

Knowledge Engineering: KB construction is not a one-time task that can be tackled by a single method in an end-to-end manner. We need to keep in mind that KBs serve as infrastructure assets and must be maintained over long timespans. Typically, we start with building an initial KB of limited scope and size, and then gradually grow it over time. This life-cycle involves tasks like KB augmentation by adding properties (Chapter 6), discovering new entities, schema expansion (Chapter 7), and most importantly, *quality assurance* and *KB curation* (Chapter 8).

For this entire framework, but also for each of its sub-tasks, a substantial amount of engineering is required, with humans like KB architects and KB curators in the loop. KB construction relies on clever algorithms and smart machine learning, but orchestrating and steering the complete machinery cannot be fully automated.

Diverse Toolbox: For many sub-tasks of KB creation and curation, we have presented a variety of alternative methods. Some readers may have hoped for a clear recommendation of a single best-practice choice, but this is unrealistic. There are many trade-offs (e.g., precision-recall-cost) and dependencies on application requirements. With varying sweet spots and limitations of different algorithms and learners, there is no one-size-fits-all, turn-key method (and hoping for this by more AI advances is likely wishful thinking).

Put positively, a wide portfolio of different models, methods and tools for knowledge extraction and knowledge cleaning is a great asset to build on. Each sub-task in the KB life-cycle mandates judicious choices from this toolbox, again emphasizing the need for humans in the loop. For this reason, we have tried to cover a diverse variety of approaches throughout this (consequently fairly long) article.

10.2 Challenges and Opportunities

In this final section, we sketch some of the open challenges that remain to be addressed towards the next generation of knowledge bases, pointing out opportunities for original and potentially impactful research.

Language Models and Knowledge Bases:
Many methods for knowledge extraction harness different kinds of language models, from lexicons of word senses and lexical relations [159, 203] all the way to contextualized embeddings such as ELMo, BERT and GPT-3 [448, 118, 62]. As the latter encode huge text corpora and are trained to predict missing words or phrases, an intriguing idea could be to use *neural language models* directly as a proxy KB (see, e.g., [449, 269]). For example, instead of looking up which river flows through Paris, we could ask a neural model to predict the word or phrase "[?]" in the incomplete sentence

"Paris is located on the banks of the [?]",

returning "Seine" as best prediction (followed by Loire, Danube, Mississippi etc.). However, when we want to obtain protest songs by Bob Dylan from

"Dylan wrote protest songs such as [?]",

the top prediction is "this", and the predictions for

"Dylan wrote [?] songs such as Hurricane"

are "popular", "other", "many", "several", "three" etc, but not "protest" or "political". These results are sort of correct, but completely useless.

Nevertheless, the impressive capabilities of these models for reading comprehension could be leveraged more extensively for KB construction and augmentation. The interplay of symbolic knowledge and latent understanding of language is an important research avenue [468].

Credibility and Trust:
We live in times with online information exploding in volume, velocity and varying shades of veracity. Unfortunately, this includes a large fraction of (false) misinformation and (intentionally false) disinformation. To support people in detecting fake news and assessing the credibility of claims in social media, research on fact checking and propaganda detection has become a major direction (e.g., [474, 459, 583, 662, 1, 371, 268]). When KB construction taps into riskier sources such as polarized news sources or discussion forums, assessing credibility and trustworthiness becomes crucial. Conversely, KB statements and reasoning over them could potentially contribute also to unmasking false claims.

An important use case is the domain of health. On one hand, health communities and specific discussion forums are sources for learning more about symptoms of diseases, typical (as opposed to potentially possible) side effects of drugs, the patients' experience with therapies, and all kinds of cross-talk signals about combinations of medical treatments. On the other hand, such social media come with a large amount of noisy and false statements (see, e.g., [408]). Contemporary examples are treatments for virus-caused epidemic diseases and the pros and cons of vaccinations. Even scientific news are often under debate (e.g., [540, 606]). Tackling these concerns about veracity is an inherent part of knowledge acquistion for the health domain.

Commonsense and Socio-Cultural Knowledge:
Commonsense knowledge (CSK) is the AI term for non-encyclopedic world knowledge that virtually all humans agree on. This comprises:

- Notable *properties of everyday objects*, such as: mountains are high and steep, they are taller than man-made buildings, they may be snow-covered or rocky (but they are never fast or funny); cups are usually round and are used to hold liquids such as coffee, tea etc.
- *Behavioral patterns* and *causality*, such as: children live with their parents, pregnancy leads to birth, and so on.
- *Human activities* and their typical settings, such as: concerts involve musicians, instruments and an audience; rock concerts involve amplifiers and usually take place in large concert halls or open air festivals (rather than cozy bars or dinner places).

This kind of CSK about general concepts and activities (as opposed to individual named entities) is obvious for humans: every young adult or even child knows this. However, CSK is surprisingly difficult to acquire by machines. There is ample AI work on knowledge representation to this end [113], introducing, for example, epistemic logics with modalities like `always`, `sometimes`, `typically` or `never`, but there is not much work on building large-scale CSK collections. Some works consider concept hierarchies like WordNet-style hypernymy as CSK, but this can equally be seen as part of encyclopedic KBs and has been intensively researched (see Chapters 3 and 4).

CSK may not be needed for today's mega-applications like search engines, but it could become a crucial building block for next-generation AI. An envisioned use case is to equip conversational assistants, like chatbots, with commonsense background knowledge. This would enable them to understand their human dialog partners better (e.g., humor) and behave more robustly in their generated utterances (e.g., avoiding absurdities and offensive statements). Interpreting visual contents, especially videos with speech, would be another potential use case of high importance [651].

A variation of CSK that matters for human-computer interaction is *socio-cultural knowledge*: behavioral patterns that do not necessarily hold universally, but are widely agreed upon within a large socio-cultural group. For example, people in the western world greet each other by handshakes, but this is very uncommon in large parts of Asia. Similarly, there are preferred styles of eating meals, with different utensils, different ways of sharing, etc. These are not just specific to geo-regions, but need to consider social and cultural backgrounds; for example, children and juveniles do not greet each other by handshakes even in the western world.

Various projects have applied knowledge extraction methods, like those in Chapters 6 and 7, to gather CSK statements from online contents (e.g., [576, 575, 659, 393, 510, 495, 79]), tapping into non-standard sources like book texts, online forums, frequent queries and image descriptions. A major difficulty to tackle is that mundane CSK is rarely expressed explicitly [517] (hence the need to tap non-standard sources), and when it is, this is often in a very biased, atypical or misleading manner [187]. For example, web-frequency signals would strongly suggest that most `programmers` are `lonely` and `socially awkward` (which is disproven by the readers of this article, hopefully). The most successful CSK projects so far seem to be the conservative ones that relied on human inputs by crowdsourcing, like ConceptNet [532, 548, 549], or knowledge engineers, like Cyc [322, 372], but these are fairly limited in scope and scale. There are great research opportunities to re-think these prior approaches and devise new ones to advance CSK acquisition.

Personal and Subjective Knowledge:
Another step away from encyclopedic knowledge of universal interest is to capture knowledge about individual users, like their habits, tastes and preferences. This would be based on the digital footprint that a user leaves by her online behavior, including clicks, likes, purchases etc. This kind of knowledge for *user profiling* is collected by major platforms for shopping, entertainment, social media, etc., and leveraged to personalize recommendations and services.

However, there would be big merit in providing a user with her *personal KB* also on the user's desktop and/or smartphone. This KB could serve as an "augmented memory" that assists the user by smart search over her personal digital history. For example, the user could easily retrieve restaurants where the user dined during her vacation in Italy and could recollect which places and dishes she liked best. Research on this notion of a personal KB dates back to the theme of "Personal Information Management" (e.g., [139, 132, 122]) and is being revived now (e.g., [398, 28]).

In addition to such services for end-users, there is an even more compelling reason for personal KBs under user control, namely, privacy and, more specifically, the need for *inverse privacy* [202]: knowing what others (the big providers) know about you! Although modern legislation such as the EU-wide GDPR (General Data Protection Regulation) mandate that users have the right to inquire provider-side profiling and have unwanted information removed, the implementation is all but easy. Users leave their data in a wide variety of web and cloud services over a very long timeframe, and most will simply forget which traces they have left where and when. A digital assistant could intercept the user's online access and record personal information locally, this way maintaining a longitudinal personal KB, under the control of the user herself (see, e.g., [274]). A practically viable solution still faces research challenges, though: what to capture and how to represent it, how to contextualize statements for user interpretability, how to empower end-users to manage the life-cycle of their personal KBs, and more.

While personal knowledge is still factual but refers only to one individual, there are also use cases for capturing *subjective knowledge*, regarding beliefs and argumentations. For example, it could be of interest to have statements of the form:

```
Fabian believes id1: (Elvis livesOn Mars)
Gerhard believes id2: (Dylan deserves PeaceNobelPrize)
Luna supports Gerhard on id2
Simon opposes Gerhard on id2
```

These kinds of statements, with second-order predicates about attribution and stance, are essential to capture people's positions and potential biases in discussions and controversies. Obviously, the stakeholders of interest could be important people such as Francois Macron or Angela Merkel rather than the above insignificant ones. Use cases include credibility analysis (see point *Credibility and Trust* above), argumentation mining, analyzing debates, political opinion mining, uncovering propaganda and other manipulation, reasoning over legal texts, and more (see, e.g., [21, 552, 205, 86, 431, 558, 42, 280]). How to automatically capture this kind of subjective knowledge for systematic organization in a KB is a widely open research area.

Entities with Quantities:
KBs should also support knowledge workers like (data) journalists, (business and media) analysts, health experts, and more. Such advanced users go beyond finding entities or looking up their properties, and often desire to filter, compare, group, aggregate and rank entities based on *quantities*: financial, physical, technological, medical and other measures, such as annual revenue or estimated worth, distance or speed, energy consumption or CO2 footprint, blood lab values or drug dosages. Examples of quantity-centric information needs are:

- Which runners have ten or more marathons under 2:10:00 hours?
- How do the sales/downloads, earnings and wealth of male and female singers compare?
- How do Japanese electric cars compare to US-made models, in terms of energy efficiency, carbon footprint and cost/km?

These kinds of analyses would be straightforward, using SQL or SPARQL queries and data-science tools, if the underlying data were stored in a single database or knowledge base. Unfortunately, this is not the case. KBs are notoriously sparse regarding quantities; for example, Wikidata contains several thousand marathon runners but knows their best times only for a few tens (not to speak of all their races). Instead of a KB, we could turn to domain-specific databases on the web, but finding the right sources in an "ocean of data" and assessing their quality, freshness and completeness is itself a big challenge.

Extracting statements about quantities, from text or tables, poses major issues [512, 253, 235, 619]:

i) detecting and normalizing quantities that appear with varying values (e.g., estimated or stale), with different scales (e.g., with modifiers "thousand", "K" or "Mio") and units (e.g., MPG-e (miles per gallon equivalent) vs. kWh/100km),
ii) inferring to which entity and measure a quantity mention refers, and
iii) contextualizing entity-quantity pairs with enough data for proper interpretation in downstream analytics – all this with very high quality and coverage.

Knowledge Base Coverage:
Commonsense and quantity knowledge are two dimensions on which today's KBs fall short. In general, coverage is a major concern even for very large KBs. Typically, they cover basic facts about people, places and products very well, but often miss out on more sophisticated, highly notable points such as:

- Johnny Cash performed a free concert in a prison (in the 1960s!).
- The Shoshone woman Sacagawea carried her newborn child when serving as guide and interpreter for the Lewis and Clark expedition.
- Frida Kahlo, the surrealistic Mexican painter, suffered her whole life from injuries in a bus accident.
- Cixin Liu's book Three Body features locations like Tsinghua University and Alpha Centauri.

These facts are prominently mentioned in the respective Wikipedia articles and would be remembered as salient points by most readers, yet they are completely absent in today's KBs. Despite advances on Open IE (Chapter 7) to discover new predicates, coverage is a major pain point. This calls for re-thinking the notion of knowledge saliency and the approaches to capture "unknown unknowns".

References

[1] B. Adair, C. Li, J. Yang, and C. Yu. "Automated pop-up fact-checking: Challenges & progress". In: *Computation + Journalism Symposium*. 2019.

[2] D. Adiwardana, M. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, and Q. V. Le. "Towards a Human-like Open-Domain Chatbot". *CoRR*. abs/2001.09977. 2020.

[3] E. Agichtein and L. Gravano. "*Snowball*: extracting relations from large plain-text collections". In: *Conference on Digital Libraries*. 2000.

[4] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. "Fast Discovery of Association Rules". In: *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996. 307–328.

[5] R. Agrawal and R. Srikant. "Fast Algorithms for Mining Association Rules in Large Databases". In: *Conference on Very Large Databases (VLDB)*. 1994.

[6] S. Akasaki, N. Yoshinaga, and M. Toyoda. "Early Discovery of Emerging Entities in Microblogs". In: *Joint Conference on Artificial Intelligence (IJCAI)*. 2019.

[7] J. Alammar. "The Illustrated Transformer". *http://jalammar.github.io/illustrated-transformer*. 2018.

[8] O. Alonso. *The Practice of Crowdsourcing*. Morgan & Claypool Publishers, 2019.

[9] O. Alonso and T. Sellam. "Quantitative Information Extraction From Social Data". In: *ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 2018.

[10] C. Alt, M. Hübner, and L. Hennig. "Fine-tuning Pre-Trained Transformer Language Models to Distantly Supervised Relation Extraction". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019.

[11] R. A. Al-Zaidy and C. L. Giles. "Extracting Semantic Relations for Scholarly Knowledge Base Construction". In: *International Conference on Semantic Computing (ICSC)*. 2018.

[12] W. Ammar, D. Groeneveld, C. Bhagavatula, I. Beltagy, M. Crawford, D. Downey, J. Dunkelberger, A. Elgohary, S. Feldman, V. Ha, *et al.* "Construction of the literature graph in semantic scholar". *arXiv:1805.02262*. 2018.

[13] G. Angeli, M. J. J. Premkumar, and C. D. Manning. "Leveraging Linguistic Structure For Open Domain Information Extraction". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2015.

[14] M. Arenas, L. E. Bertossi, and J. Chomicki. "Consistent Query Answers in Inconsistent Databases". In: *ACM Symposium on Principles of Database Systems (PODS)*. 1999.

[15] A. G. Arens-Volland, B. Gâteau, and Y. Naudet. “Semantic Modeling for Personalized Dietary Recommendation”. In: *IEEE Workshop on Semantic and Social Media Adaptation and Personalization*. 2018.

[16] H. Arnaout, S. Razniewski, and G. Weikum. “Enriching Knowledge Bases with Interesting Negative Statements”. In: *Conference on Automatic Knowledge Base Construction (AKBC)*. 2020.

[17] A. R. Aronson and F. Lang. “An overview of MetaMap: historical perspective and recent advances”. *J. Am. Medical Informatics Assoc.* 17(3): 229–236. 2010.

[18] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. “DBpedia: A Nucleus for a Web of Open Data”. In: *International Semantic Web Conference (ISWC)*. 2007.

[19] S. Auer and J. Lehmann. “What Have Innsbruck and Leipzig in Common? Extracting Semantics from Wiki Content”. In: *European Semantic Web Conference (ESWC)*. 2007.

[20] D. Aumueller, H. H. Do, S. Massmann, and E. Rahm. “Schema and ontology matching with COMA++”. In: *ACM Conference on Management of Data (SIGMOD)*. 2005.

[21] R. Awadallah, M. Ramanath, and G. Weikum. “Harmony and dissonance: organizing the people’s voices on political controversies”. In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2012.

[22] F. Baader, I. Horrocks, C. Lutz, and U. Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

[23] S. H. Bach, M. Broecheler, B. Huang, and L. Getoor. “Hinge-Loss Markov Random Fields and Probabilistic Soft Logic”. *Journal of Machine Learning Research (JMLR)*. 18: 109:1–109:67. 2017.

[24] R. A. Baeza-Yates and A. Tiberi. “Extracting semantic relations from query logs”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2007.

[25] A. Bagga and B. Baldwin. “Entity-Based Cross-Document Coreferencing Using the Vector Space Model”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 1998.

[26] Baidu. “Introducing Qian Yan, Baidu’s New Plan to Build 100 Chinese NLP Datasets in Three Years”. In: *Baidu Research Blog, 28 August 2020, http://research.baidu.com/Blog/index-view?id=146*.

[27] V. Balaraman, S. Razniewski, and W. Nutt. “Recoin: Relative Completeness in Wikidata”. In: *Companion of the The Web Conference*. 2018.

[28] K. Balog and T. Kenter. “Personal Knowledge Graphs: A Research Agenda”. In: *ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 2019.

[29] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. “Open Information Extraction from the Web”. In: *Joint Conference on Artificial Intelligence (IJCAI)*. 2007.

[30] C. J. Bannard and C. Callison-Burch. “Paraphrasing with Bilingual Parallel Corpora”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2005.

[31] M. Bansal, D. Burkett, G. de Melo, and D. Klein. “Structured Learning for Taxonomy Induction with Belief Propagation”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2014. 1041–1051.

[32] T. Bansal, P. Verga, N. Choudhary, and A. McCallum. “Simultaneously Linking Entities and Extracting Relations from Biomedical Text Without Mention-level Supervision”. *CoRR*. abs/1912.01070. 2019.

[33] L. Barbosa and J. Freire. “An adaptive crawler for locating hidden web entry points”. In: *The Web Conference (WWW)*. 2007.

[34] R. Barzilay and K. R. McKeown. “Extracting Paraphrases from a Parallel Corpus”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2001.

[35] H. Bast, B. Buchhold, and E. Haussmann. “Semantic Search on Text and Knowledge Bases”. *Foundations and Trends in Information Retrieval*. 10(2-3): 119–271. 2016.

[36] R. Baumgartner, S. Flesca, and G. Gottlob. “Visual Web Information Extraction with Lixto”. In: *Conference on Very Large Databases (VLDB)*. 2001.

[37] L. E. Bertossi. *Database Repairing and Consistent Query Answering*. Morgan & Claypool Publishers, 2011.

[38] L. E. Bertossi. “Database Repairs and Consistent Query Answering: Origins and Further Developments”. In: *ACM Symposium on Principles of Database Systems (PODS)*. 2019.

[39] C. S. Bhagavatula, T. Noraset, and D. Downey. “TabEL: Entity Linking in Web Tables”. In: *International Semantic Web Conference (ISWC)*. 2015.

[40] I. Bhattacharya and L. Getoor. “Collective entity resolution in relational data”. *ACM Transactions on Knowledge Discovery from Data (TKDD)*. 1(1): 5. 2007.

[41] N. Bhutani, H. V. Jagadish, and D. R. Radev. “Nested Propositions in Open Information Extraction”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2016.

[42] N. Bhutani, A. Traylor, C. Chen, X. Wang, B. Golshan, and W.-C. Tan. “Sampo: Unsupervised Knowledge Base Construction for Opinions and Implications”. In: *Automatic Knowledge Base Construction (AKBC)*. 2020.

[43] J. A. Biega, E. Kuzey, and F. M. Suchanek. “ Inside YAGO2s: A Transparent Information Extraction Architecture ”. In: *The Web Conference (WWW)*. 2013.

[44] M. Bienvenu. “A Short Survey on Inconsistency Handling in Ontology-Mediated Query Answering”. *KI-Künstliche Intelligenz*: 1–9. 2020.

[45] *Handbook of Satisfiability*. Vol. 185. *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[46] C. Bizer, K. Eckert, R. Meusel, H. Mühleisen, M. Schuhmacher, and J. Völker. "Deployment of RDFa, Microdata, and Microformats on the Web - A Quantitative Analysis". In: *International Semantic Web Conference (ISWC)*. 2013.

[47] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. "DBpedia - A crystallization point for the Web of Data". *Journal of Web Semantics (JWS)*. 7(3): 154–165. 2009.

[48] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent Dirichlet Allocation". *Journal of Machine Learning Research (JMLR)*. 3: 993–1022. 2003.

[49] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. "A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification". In: *ACM Conference on Management of Data (SIGMOD)*. 2005.

[50] P. Bohannon, S. Merugu, C. Yu, V. Agarwal, P. DeRose, A. S. Iyer, A. Jain, V. Kakade, M. Muralidharan, R. Ramakrishnan, and W. Shen. "Purple SOX extraction management system". *SIGMOD Record*. 37(4): 21–27. 2008.

[51] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. "Freebase: a collaboratively created graph database for structuring human knowledge". In: *ACM Conference on Management of Data (SIGMOD)*. 2008.

[52] I. Boneva, J. E. L. Gayo, and E. G. Prud'hommeaux. "Semantics and Validation of Shapes Schemas for RDF". In: *International Semantic Web Conference (ISWC)*. 2017.

[53] G. Bordea, E. Lefever, and P. Buitelaar. "Semeval-2016 task 13: Taxonomy extraction evaluation". In: *Workshop on Semantic Evaluation (SemEval)*. 2016.

[54] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. "Translating Embeddings for Modeling Multi-relational Data". In: *Neural Information Processing Systems (NeurIPS)*. 2013.

[55] Z. Bouraoui, J. Camacho-Collados, and S. Schockaert. "Inducing Relational Knowledge from BERT". *CoRR*. abs/1911.12753. 2019.

[56] C. D. Bovi, L. Telesca, and R. Navigli. "Large-Scale Information Extraction from Textual Definitions through Deep Syntactic and Semantic Analysis". *Transactions of the Association for Computational Linguistics (TACL)*. 3: 529–543. 2015.

[57] A. Brack, A. Hoppe, M. Stocker, S. Auer, and R. Ewerth. "Requirements Analysis for an Open Research Knowledge Graph". In: *International Conference on Theory and Practice of Digital Libraries (TPDL)*. 2020.

[58] S. Brin. "Extracting Patterns and Relations from the World Wide Web". In: *International Workshop on the Web and Databases (WebDB)*. 1998.

[59] S. Brin and L. Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". *Computer Networks*. 30(1-7): 107–117. 1998.

[60] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. "Syntactic Clustering of the Web". *Computer Networks*. 29(8-13): 1157–1166. 1997.

[61] M. Bronzi, V. Crescenzi, P. Merialdo, and P. Papotti. "Extraction and Integration of Partially Overlapping Web Sources". *Proceedings of the VLDB Endowment*. 6(10): 805–816. 2013.

[62] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. "Language Models are Few-Shot Learners". *arXiv: 2005.14165*. 2020.

[63] P. Buneman, S. Khanna, and W.-C. Tan. "Why and Where: A Characterization of Data Provenance". In: *International Conference on Database Theory (ICDT)*. 2001.

[64] P. Buneman and W.-C. Tan. "Data Provenance: What next?" *SIGMOD Record*. 47(3): 5–16. 2018.

[65] R. C. Bunescu and R. J. Mooney. "A Shortest Path Dependency Kernel for Relation Extraction". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2005.

[66] R. C. Bunescu and R. J. Mooney. "Subsequence Kernels for Relation Extraction". In: *Neural Information Processing Systems (NeurIPS)*. 2005.

[67] R. C. Bunescu and M. Pasca. "Using Encyclopedic Knowledge for Named entity Disambiguation". In: *European Chapter of the Association for Computational Linguistics (EACL)*. 2006.

[68] A. Burkov. *The Hundred-Page Machine Learning Book*. 2019.

[69] M. J. Cafarella, A. Y. Halevy, H. Lee, J. Madhavan, C. Yu, D. Z. Wang, and E. Wu. "Ten Years of WebTables". *Proceedings of the VLDB Endowment*. 11(12): 2140–2149. 2018.

[70] D. Cai, S. Yu, J. Wen, and W. Ma. "Block-based web search". In: *ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 2004.

[71] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. "Toward an Architecture for Never-Ending Language Learning". In: *Conference on Artificial Intelligence (AAAI)*. 2010.

[72] A. Carlson, J. Betteridge, R. C. Wang, E. R. H. Jr., and T. M. Mitchell. "Coupled semi-supervised learning for information extraction". In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2010.

[73] G. Carothers. "RDF 1.1 N-Quads, https://www.w3.org/TR/2014/REC-n-quads-20140225/". *W3C Recommendation*. W3C, 2014.

[74] R. Caruana and A. Niculescu-Mizil. “An empirical comparison of supervised learning algorithms”. In: *International Conference on Machine Learning (ICML)*. 2006.

[75] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani. “Learning relatedness measures for entity linking”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2013.

[76] K. Chakrabarti, S. Chaudhuri, Z. Chen, K. Ganjam, and Y. He. “Data services leveraging Bing’s data assets”. *IEEE Data Engineering Bulletin*. 39(3): 15–28. 2016.

[77] S. Chakrabarti. *Mining the web - discovering knowledge from hypertext data*. Morgan Kaufmann, 2003.

[78] S. Chakrabarti, M. van den Berg, and B. Dom. “Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery”. *Computer Networks*. 31(11-16): 1623–1640. 1999.

[79] Y. Chalier, S. Razniewski, and G. Weikum. “Joint Reasoning for Multi-Faceted Commonsense Knowledge”. *International Conference on Automatic Knowledge Base Construction (AKBC)*. 2020.

[80] K. Chang, W. Yih, B. Yang, and C. Meek. “Typed Tensor Decomposition of Knowledge Bases for Relation Extraction”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[81] M. Chang, L. Ratinov, and D. Roth. “Structured learning with constrained conditional models”. *Machine Learning*. 88(3): 399–431. 2012.

[82] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. “Robust and Efficient Fuzzy Match for Online Data Cleaning”. In: *ACM Conference on Management of Data (SIGMOD)*. 2003.

[83] H. Chen, M. Vasardani, and S. Winter. “Clustering-based disambiguation of fine-grained place names from descriptions”. *GeoInformatica*. 23(3): 449–472. 2019.

[84] J. Chen, X. Chen, I. Horrocks, E. B. Myklebust, and E. Jiménez-Ruiz. “Correcting Knowledge Base Assertions”. In: *The Web Conference (WWW)*. 2020.

[85] J. Chen, E. Jiménez-Ruiz, I. Horrocks, and C. A. Sutton. “ColNet: Embedding the Semantics of Web Tables for Column Type Prediction”. In: *Conference on Artificial Intelligence (AAAI)*. 2019.

[86] S. Chen, D. Khashabi, W. Yin, C. Callison-Burch, and D. Roth. “Seeing Things from a Different Angle: Discovering Diverse Perspectives about Claims”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2019.

[87] Z. Chen, M. J. Cafarella, and H. V. Jagadish. “Long-tail Vocabulary Dictionary Extraction from the Web”. In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2016. 625–634.

[88] F. Chiang and R. J. Miller. “Discovering data quality rules”. *Proceedings of the VLDB Endowment*. 1(1): 1166–1177. 2008.

[89] L. Chiticariu, M. Danilevsky, Y. Li, F. Reiss, and H. Zhu. “SystemT: Declarative Text Understanding for Enterprise”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[90] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, and S. Vaithyanathan. “SystemT: An Algebraic Approach to Declarative Information Extraction”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2010.

[91] A. I. Chittilappilly, L. Chen, and S. Amer-Yahia. “A Survey of General-Purpose Crowdsourcing Techniques”. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 28(9): 2246–2266. 2016.

[92] T. Chklovski and P. Pantel. “VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2004.

[93] E. Choi, O. Levy, Y. Choi, and L. Zettlemoyer. “Ultra-Fine Entity Typing”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2018.

[94] P. Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer, 2012.

[95] C. Christodoulopoulos and A. Mittal. “Simple Large-scale Relation Extraction from Unstructured Text”. In: *Conference on Language Resources and Evaluation (LREC)*. 2018.

[96] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, and K. Stefanidis. “End-to-End Entity Resolution for Big Data: A Survey”. *CoRR*. abs/1905.06397. 2019.

[97] C. X. Chu, S. Razniewski, and G. Weikum. “TiFi: Taxonomy Induction for Fictional Domains”. In: *The Web Conference (WWW)*. 2019.

[98] C. X. Chu, N. Tandon, and G. Weikum. “Distilling Task Knowledge from How-To Communities”. In: *The Web Conference (WWW)*. 2017.

[99] X. Chu, I. F. Ilyas, and P. Papotti. “Holistic data cleaning: Putting violations into context”. In: *IEEE International Conference on Data Engineering (ICDE)*. 2013.

[100] K. Clark and C. D. Manning. “Improving Coreference Resolution by Learning Entity-Level Distributed Representations”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2016.

[101] L. D. Corro and R. Gemulla. “ClausIE: clause-based open information extraction”. In: *The Web Conference (WWW)*. 2013.

[102] A. Cropper, S. Dumancic, and S. H. Muggleton. “Turning 30: New Ideas in Inductive Logic Programming”. *CoRR*. abs/2002.11002. 2020.

[103] S. Cucerzan. “Large-Scale Named Entity Disambiguation Based on Wikipedia Data”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2007.

[104] S. Cucerzan and A. Sil. “The MSR Systems for Entity Linking and Temporal Slot Filling at TAC 2013”. In: *Text Analysis Conference (TAC)*. 2013.

[105] A. Culotta and J. S. Sorensen. “Dependency Tree Kernels for Relation Extraction”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2004.

[106] H. Cunningham. “GATE, a General Architecture for Text Engineering”. *Computers and the Humanities*. 36(2): 223–254. 2002.

[107] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. “A framework and graphical development environment for robust NLP tools and applications”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2002.

[108] J. Dai, M. Zhang, G. Chen, J. Fan, K. Y. Ngiam, and B. C. Ooi. “Fine-grained Concept Linking using Neural Networks in Healthcare”. In: *ACM Conference on Management of Data (SIGMOD)*. ACM, 2018.

[109] B. B. Dalvi, W. W. Cohen, and J. Callan. “WebSets: extracting sets of entities from the web using unsupervised information extraction”. In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2012.

[110] N. N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. S. Keerthi, and S. Merugu. “A web of concepts”. In: *ACM Symposium on Principles of Database Systems (PODS)*. 2009.

[111] F. Darari, W. Nutt, G. Pirrò, and S. Razniewski. “Completeness Management for RDF Data Sources”. *ACM Transactions on the Web (TWEB)*. 12(3): 18:1–18:53. 2018.

[112] R. Das, T. Munkhdalai, X. Yuan, A. Trischler, and A. McCallum. “Building Dynamic Knowledge Graphs from Text using Machine Reading Comprehension”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[113] E. Davis. *Representations of commonsense knowledge*. Morgan Kaufmann, 2014.

[114] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. “Indexing by Latent Semantic Analysis”. *JASIS*. 41(6): 391–407. 1990.

[115] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. “ImageNet: A large-scale hierarchical image database”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009.

[116] L. Derczynski, D. Maynard, G. Rizzo, M. van Erp, G. Gorrell, R. Troncy, J. Petrak, and K. Bontcheva. “Analysis of named entity recognition and linking for tweets”. *Information Processing and Management*. 51(2): 32–49. 2015.

[117] O. Deshpande, D. S. Lamba, M. Tourn, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. “Building, maintaining, and using knowledge bases: a report from the Trenches”. In: *ACM Conference on Management of Data (SIGMOD)*. 2013.

[118] J. Devlin, M. Chang, K. Lee, and K. Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2019.

[119] D. Diefenbach, V. López, K. D. Singh, and P. Maret. “Core techniques of question answering systems over knowledge bases: a survey”. *Knowledge and Information Systems (KAIS)*. 55(3): 529–569. 2018.

[120] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. V. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. “SemTag and seeker: bootstrapping the semantic web via automated semantic annotation”. In: *The Web Conference (WWW)*. 2003.

[121] J. Dittrich *et al.* “Research Report - Use of Wikidata in GLAM Institutions, https://commons.wikimedia.org/wiki/File:Research_Report_\T1\textendash_Use_of_Wikidata_in_GLAM_institutions_(2019-11).pdf”. *Wikimedia Deutschland e.V.* 2019.

[122] J. Dittrich, M. A. V. Salles, D. Kossmann, and L. Blunschi. “iMeMex: Escapes from the Personal Information Jungle”. In: *Conference on Very Large Databases (VLDB)*. 2005.

[123] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[124] A. Doan, R. Ramakrishnan, and A. Y. Halevy. “Crowdsourcing systems on the World-Wide Web”. *Commun. ACM*. 54(4): 86–96. 2011.

[125] P. M. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool Publishers, 2009.

[126] P. M. Domingos and D. Lowd. “Unifying logical and statistical AI with Markov logic”. *Communication of the ACM*. 62(7): 74–83. 2019.

[127] X. L. Dong. “Building a Broad Knowledge Graph for Products”. In: *Slides from Keynote at International Conference on Data Engineering (ICDE), http://lunadong.com/talks/BG.pdf*. 2019.

[128] X. L. Dong, L. Berti-Équille, and D. Srivastava. “Data Fusion: Resolving Conflicts from Multiple Sources”. In: *Handbook of Data Quality, Research and Practice*. 2013. 293–318.

[129] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. “Knowledge vault: a web-scale approach to probabilistic knowledge fusion”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2014.

[130] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang. “From Data Fusion to Knowledge Fusion”. *Proceedings of the VLDB Endowment*. 7(10): 881–892. 2014.

[131] X. L. Dong, E. Gabrilovich, K. Murphy, V. Dang, W. Horn, C. Lugaresi, S. Sun, and W. Zhang. "Knowledge-Based Trust: Estimating the Trustworthiness of Web Sources". *Proceedings of the VLDB Endowment*. 8(9): 938–949. 2015.

[132] X. L. Dong and A. Y. Halevy. "A Platform for Personal Information Management and Integration". In: *Conference on Innovative Data Systems Research (CIDR)*. 2005.

[133] X. L. Dong, X. He, A. Kan, X. Li, Y. Liang, J. Ma, Y. E. Xu, C. Zhang, T. Zhao, G. B. Saldana, S. Deshpande, A. M. Manduca, J. Ren, S. P. Singh, F. Xiao, H. Chang, G. Karamanolakis, Y. Mao, Y. Wang, C. Faloutsos, A. McCallum, and J. Han. "AutoKnow: Self-Driving Knowledge Collection for Products of Thousands of Types". In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2020.

[134] X. L. Dong and F. Naumann. "Data fusion - Resolving Data Conflicts for Integration". *Proceedings of the VLDB Endowment*. 2(2): 1654–1655. 2009.

[135] X. L. Dong and D. Srivastava. *Big Data Integration*. Morgan & Claypool Publishers, 2015.

[136] D. M. Dooley, E. J. Griffiths, G. S. Gosal, P. L. Buttigieg, R. Hoehndorf, M. C. Lange, L. M. Schriml, F. S. Brinkman, and W. W. Hsiao. "FoodOn: a harmonized food ontology to increase global food traceability, quality control and data integration". *Science of Food*. 2(1): 1–10. 2018.

[137] E. C. Dragut, W. Meng, and C. T. Yu. *Deep Web Query Interface Understanding and Integration*. Morgan & Claypool Publishers, 2012.

[138] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. "Entity Disambiguation for Knowledge Base Population". In: *Conference on Computational Linguistics (COLING)*. 2010.

[139] S. T. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. "Stuff I've seen: a system for personal information retrieval and re-use". In: *ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 2003.

[140] H. L. Dunn. "Record linkage". *American Journal of Public Health and the Nations Health*. 36(12): 1412–1416. 1946.

[141] B. V. Durme and M. Pasca. "Finding Cars, Goddesses and Enzymes: Parametrizable Acquisition of Labeled Instances for Open-Domain Information Extraction". In: *Conference on Artificial Intelligence (AAAI)*. 2008.

[142] G. Durrett and D. Klein. "A Joint Model for Entity Analysis: Coreference, Typing, and Linking". *Transactions of the Association for Computational Linguistics (TACL)*. 2: 477–490. 2014.

[143] G. Durrett and D. Klein. "Easy Victories and Uphill Battles in Coreference Resolution". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2013.

[144] S. Dutta and G. Weikum. “C3EL: A Joint Model for Cross-Document Co-Reference Resolution and Entity Linking”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2015.

[145] M. Ehrmann, F. Cecconi, D. Vannella, J. P. McCrae, P. Cimiano, and R. Navigli. “Representing Multilingual Data as Linked Data: the Case of BabelNet 2.0”. In: *Conference on Language Resources and Evaluation (LREC)*. 2014.

[146] J. Eisenstein. *Introduction to Natural Language Processing*. MIT Press, 2019.

[147] D. Erdös and P. Miettinen. “Discovering facts with boolean tensor tucker decomposition”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2013.

[148] P. Ernst, A. Siu, D. Milchevski, J. Hoffart, and G. Weikum. “DeepLife: An Entity-aware Search, Analytics and Exploration Platform for Health and Life Sciences”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2016.

[149] P. Ernst, A. Siu, and G. Weikum. “HighLife: Higher-arity Fact Harvesting”. In: *The Web Conference (WWW)*. 2018.

[150] P. Ernst, A. Siu, and G. Weikum. “KnowLife: a versatile approach for constructing a large knowledge graph for biomedical sciences”. *BMC Bioinformatics*. 16: 157:1–157:13. 2015.

[151] Y. Eshel, N. Cohen, K. Radinsky, S. Markovitch, I. Yamada, and O. Levy. “Named Entity Disambiguation for Noisy Text”. In: *Conference on Computational Natural Language Learning (CoNLL)*. 2017.

[152] O. Etzioni, M. Banko, and M. J. Cafarella. “Machine Reading”. In: *Conference on Artificial Intelligence (AAAI)*. 2006.

[153] O. Etzioni, M. J. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. “Unsupervised named-entity extraction from the Web: An experimental study”. *Artificial Intelligence (AI)*. 165(1): 91–134. 2005.

[154] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. “Open Information Extraction: The Second Generation”. In: *Joint Conference on Artificial Intelligence (IJCAI)*. 2011.

[155] A. Fader, S. Soderland, and O. Etzioni. “Identifying relations for open information extraction”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2011.

[156] M. Färber. “The Microsoft Academic Knowledge Graph: A Linked Data Source with 8 Billion Triples of Scholarly Data”. In: *International Semantic Web Conference (ISWC)*. 2019.

[157] M. Färber, F. Bartscherer, C. Menne, and A. Rettinger. “Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO”. *Semantic Web Journal (SWJ)*. 9(1): 77–129. 2018.

[158] M. Faruqui and S. Kumar. “Multilingual Open Relation Extraction Using Cross-lingual Projection”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2015.

[159] C. Fellbaum and G. A. Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[160] I. P. Fellegi and A. B. Sunter. “A theory for record linkage”. *Journal of the American Statistical Association*. 64(328): 1183–1210. 1969.

[161] P. Ferragina and U. Scaiella. “TAGME: on-the-fly annotation of short text fragments (by wikipedia entities)”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2010.

[162] E. Ferrara, P. D. Meo, G. Fiumara, and R. Baumgartner. “Web data extraction, applications and techniques: A survey”. *Knowledge Based Systems*. 70: 301–323. 2014.

[163] D. A. Ferrucci *et al.* “Special Issue on "This is Watson"”. *IBM Journal of Research and Development*. 56(3): 1. 2012.

[164] D. A. Ferrucci and A. Lally. “UIMA: an architectural approach to unstructured information processing in the corporate research environment”. *Natural Language Engineering*. 10(3-4): 327–348. 2004.

[165] C. J. Fillmore, C. Wooters, and C. F. Baker. “Building a Large Lexical Databank Which Provides Deep Semantics”. In: *Pacific Asia Conference on Language, Information and Computation (PACLIC)*. 2001.

[166] J. R. Finkel, T. Grenager, and C. D. Manning. “Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2005.

[167] N. FitzGerald, J. Michael, L. He, and L. Zettlemoyer. “Large-Scale QA-SRL Parsing”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2018.

[168] M. Fleischman and E. H. Hovy. “Fine Grained Classification of Named Entities”. In: *Conference on Computational Linguistics (COLING)*. 2002.

[169] M. Francis-Landau, G. Durrett, and D. Klein. “Capturing Semantic Similarity for Entity Linking with Convolutional Neural Networks”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2016.

[170] C. S. Funk, W. A. B. Jr., B. Garcia, C. Roeder, M. Bada, K. B. Cohen, L. E. Hunter, and K. Verspoor. “Large-scale biomedical concept recognition: an evaluation of current automatic annotators and their parameters”. *BMC Bioinformatics*. 15: 59. 2014.

[171] E. Gabrilovich and S. Markovitch. “Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis”. In: *Joint Conference on Artificial Intelligence (IJCAI)*. 2007.

[172] M. H. Gad-Elrab, D. Stepanova, J. Urbani, and G. Weikum. "Exception-enriched rule learning from knowledge graphs". In: *International Semantic Web Conference (ISWC)*. 2016.

[173] M. H. Gad-Elrab, D. Stepanova, J. Urbani, and G. Weikum. "ExFaKT: A Framework for Explaining Facts over Knowledge Graphs and Text". In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2019.

[174] L. A. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. *AMIE: association rule mining under incomplete evidence in ontological knowledge bases*. 2013.

[175] L. Galárraga, G. Heitz, K. Murphy, and F. M. Suchanek. "Canonicalizing Open Knowledge Bases". In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2014.

[176] L. Galárraga, S. Razniewski, A. Amarilli, and F. M. Suchanek. "Predicting Completeness in Knowledge Bases". In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2017.

[177] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. "Fast rule mining in ontological knowledge bases with AMIE+". *Journal of the VLDB*. 24(6): 707–730. 2015.

[178] O. Ganea, M. Ganea, A. Lucchi, C. Eickhoff, and T. Hofmann. "Probabilistic Bag-Of-Hyperlinks Model for Entity Linking". In: *The Web Conference (WWW)*. 2016.

[179] J. Ganitkevitch, B. V. Durme, and C. Callison-Burch. "PPDB: The Paraphrase Database". In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2013.

[180] J. Gao, X. Li, Y. E. Xu, B. Sisman, X. L. Dong, and J. Yang. "Efficient Knowledge Graph Accuracy Evaluation". *Proceedings of the VLDB Endowment*. 12(11): 1679–1691. 2019.

[181] K. Gashteovski, R. Gemulla, and L. D. Corro. "MinIE: Minimizing Facts in Open Information Extraction". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017.

[182] J. E. L. Gayo, E. Prud'hommeaux, H. R. Solbrig, and J. M. A. Rodriguez. "Validating and Describing Linked Data Portals using RDF Shape Expressions." In: *Workshop on Linked Data Quality (LDQ), co-located with International Conference on Semantic Systems (SEMANTICS)*. 2014.

[183] M. Geffet and I. Dagan. "The Distributional Inclusion Hypotheses and Lexical Entailment". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2005.

[184] S. Ghosh, S. Razniewski, and G. Weikum. "Uncovering Hidden Semantics of Set Information in Knowledge Bases". *Journal of Web Semantics (JWS)*. 64. 2020.

[185] D. Gildea and D. Jurafsky. "Automatic Labeling of Semantic Roles". *Computational Linguistics*. 28(3): 245–288. 2002.

[186] Y. Goldberg. *Neural Network Methods for Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[187] J. Gordon and B. Van Durme. "Reporting bias and knowledge acquisition". In: *International Workshop on Automated Knowledge Base Construction (AKBC)*. 2013.

[188] M. R. Gormley, M. Yu, and M. Dredze. "Improved Relation Extraction with Feature-Rich Compositional Embedding Models". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2015.

[189] K. Grauman. "Computer Vision for Fashion: From Individual Recommendations to World-wide Trends (Keynote), https://www.cs.utexas.edu/~grauman/slides/wsdm2020_fashion_grauman.pptx". In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2020.

[190] T. Grütze, G. Kasneci, Z. Zuo, and F. Naumann. "CohEEL: Coherent and efficient named entity linking through random walks". *Journal of Web Semantics (JWS)*. 37-38: 75–89. 2016.

[191] A. Grycner and G. Weikum. "POLY: Mining Relational Paraphrases from Multilingual Sentences". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2016.

[192] A. Grycner, G. Weikum, J. Pujara, J. R. Foulds, and L. Getoor. "RELLY: Inferring Hypernym Relationships Between Relational Phrases". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2015.

[193] R. V. Guha, D. Brickley, and S. Macbeth. "Schema.org: evolution of structured data on the web". *Communication of the ACM*. 59(2): 44–51. 2016.

[194] P. Gulhane, A. Madaan, R. R. Mehta, J. Ramamirtham, R. Rastogi, S. Satpal, S. H. Sengamedu, A. Tengli, and C. Tiwari. "Web-scale information extraction with vertex". In: *IEEE International Conference on Data Engineering (ICDE)*. 2011.

[195] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He. "A Survey on Knowledge Graph-Based Recommender Systems". *CoRR*. abs/2003.00911. 2020.

[196] Z. Guo and D. Barbosa. "Robust Entity Linking via Random Walks". In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2014.

[197] Z. Guo and D. Barbosa. "Robust named entity disambiguation with random walks". *Semantic Web Journal (SWJ)*. 9(4): 459–479. 2018.

[198] A. Gupta, F. Piccinno, M. Kozhevnikov, M. Pasca, and D. Pighin. "Revisiting Taxonomy Induction over Wikipedia". In: *Conference on Computational Linguistics (COLING)*. 2016.

[199] M. Gupta, R. Li, Z. Yin, and J. Han. "Survey on social tagging techniques". *SIGKDD Explorations*. 12(1): 58–72. 2010.

[200] N. Gupta, S. Singh, and D. Roth. “Entity Linking via Joint Encoding of Types, Descriptions, and Context”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017.

[201] R. Gupta, A. Y. Halevy, X. Wang, S. E. Whang, and F. Wu. “Biperpedia: An Ontology for Search Applications”. *Proceedings of the VLDB Endowment*. 7(7): 505–516. 2014.

[202] Y. Gurevich and J. M. Wing. “Inverse privacy”. *Communication of the ACM*. 59(7): 38–42. 2016.

[203] I. Gurevych, J. Eckle-Kohler, and M. Matuschek. *Linked Lexical Knowledge Bases: Foundations and Applications*. Morgan & Claypool Publishers, 2016.

[204] B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J. R. Curran. “Evaluating Entity Linking with Wikipedia”. *Artificial Intelligence (AI)*. 194: 130–150. 2013.

[205] A. Y. Halevy. “The Ubiquity of Subjectivity”. *IEEE Data Engineering Bulletin*. 42(1): 6–9. 2019.

[206] A. Y. Halevy, M. J. Franklin, and D. Maier. “Principles of dataspace systems”. In: *ACM Symposium on Principles of Database Systems (PODS)*. 2006.

[207] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann, 2011.

[208] X. Han, T. Gao, Y. Lin, H. Peng, Y. Yang, C. Xiao, Z. Liu, P. Li, M. Sun, and J. Zhou. “More Data, More Relations, More Context and More Openness: A Review and Outlook for Relation Extraction”. *CoRR*. abs/2004.03186. 2020.

[209] M. F. Hanafi, A. Abouzied, L. Chiticariu, and Y. Li. “SEER: Auto-Generating Information Extraction Rules from User-Specified Examples”. In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 2017.

[210] Q. Hao, R. Cai, Y. Pang, and L. Zhang. “From one tree to a forest: a unified solution for structured web data extraction”. In: *ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 2011.

[211] S. Harris and A. Seaborne. “SPARQL 1.1 Query Language, https://www.w3.org/TR/2013/REC-sparql11-query-20130321/”. *W3C Recommendation*. W3C, 2013.

[212] O. Hartig. “Foundations of RDF⋆ and SPARQL⋆ (An Alternative Approach to Statement-Level Metadata in RDF)”. In: *Alberto Mendelzon Workshop (AMW)*. 2017.

[213] O. Hartig. “Provenance Information in the Web of Data.” In: *Workshop on Linked Data on the Web (LDOW), co-located with the Web Conference*. 2009.

[214] O. Hartig. “RDF* and SPARQL*: An Alternative Approach to Annotate Statements in RDF”. In: *International Semantic Web Conference (ISWC)*. 2017.

[215] Harvard NLP Group. “The Annotated Transformer”. *https://nlp.seas.harvard.edu/2018/04/03/attention.html*. 2018.

[216] A.-W. Harzing and S. Alakangas. “Google Scholar, Scopus and the Web of Science: a longitudinal and cross-disciplinary comparison”. *Scientometrics*. 106(2): 787–804. 2016.

[217] K. Hashimoto, M. Miwa, Y. Tsuruoka, and T. Chikayama. “Simple Customization of Recursive Neural Networks for Semantic Relation Classification”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2013.

[218] S. Haussmann, O. Seneviratne, Y. Chen, Y. Ne’eman, J. Codella, C. Chen, D. L. McGuinness, and M. J. Zaki. “FoodKG: A Semantics-Driven Knowledge Graph for Food Recommendation”. In: *International Semantic Web Conference (ISWC)*. 2019.

[219] T. H. Haveliwala. “Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search”. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 15(4): 784–796. 2003.

[220] L. He, K. Lee, M. Lewis, and L. Zettlemoyer. “Deep Semantic Role Labeling: What Works and What’s Next”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2017.

[221] L. He, M. Lewis, and L. Zettlemoyer. “Question-Answer Driven Semantic Role Labeling: Using Natural Language to Annotate Natural Language”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2015.

[222] Y. He, K. Chakrabarti, T. Cheng, and T. Tylenda. “Automatic Discovery of Attribute Synonyms Using Query Logs and Table Corpora”. In: *The Web Conference (WWW)*. 2016.

[223] Y. He and D. Xin. “SEISA: set expansion by iterative similarity aggregation”. In: *The Web Conference (WWW)*. 2011. 427–436.

[224] M. A. Hearst. “Automatic Acquisition of Hyponyms from Large Text Corpora”. In: *Conference on Computational Linguistics (COLING)*. 1992.

[225] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool Publishers, 2011.

[226] S. Heindorf, M. Potthast, B. Stein, and G. Engels. “Vandalism detection in wikidata”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2016.

[227] N. Heist, S. Hertling, D. Ringler, and H. Paulheim. “Knowledge Graphs on the Web - An Overview”. In: *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*. IOS Press, 2020. 3–22.

[228] N. Heist and H. Paulheim. “Entity Extraction from Wikipedia List Pages”. In: *European Semantic Web Conference (ESWC)*. 2020.

[229] S. Hellmann, C. Stadler, J. Lehmann, and S. Auer. “DBpedia Live Extraction”. In: *OTM Conferences*. 2009.

[230] D. Hernández, A. Hogan, and M. Krötzsch. “Reifying RDF: What Works Well With Wikidata?” In: *Scalable Semantic Web Knowledge Base Systems*. 2015.

[231] S. Hertling and H. Paulheim. “DBkWik: A Consolidated Knowledge Graph from Thousands of Wikis”. In: *International Conference on Big Knowledge (ICBK)*. 2018.

[232] S. Hertling and H. Paulheim. “WebIsALOD: Providing Hypernymy Relations Extracted from the Web as Linked Open Data”. In: *International Semantic Web Conference (ISWC)*. 2017.

[233] P. Heymann and H. Garcia-Molina. “Collaborative creation of communal hierarchical taxonomies in social tagging systems”. *Stanford University, Technical Report*. 2006.

[234] D. S. Himmelstein, A. Lizee, C. Hessler, L. Brueggeman, S. L. Chen, D. Hadley, A. Green, P. Khankhanian, and S. E. Baranzini. “Systematic integration of biomedical knowledge prioritizes drugs for repurposing”. *Elife*. 6. 2017.

[235] V. T. Ho, Y. Ibrahim, K. Pal, K. Berberich, and G. Weikum. “Qsearch: Answering Quantity Queries from Text”. In: *International Semantic Web Conference (ISWC)*. 2019.

[236] J. Hoffart, Y. Altun, and G. Weikum. “Discovering emerging entities with ambiguous names”. In: *The Web Conference (WWW)*. 2014.

[237] J. Hoffart, D. Milchevski, G. Weikum, A. Anand, and J. Singh. “The Knowledge Awakens: Keeping Knowledge Bases Fresh with Emerging Entities”. In: *Companion of the International Conference on World Wide Web*. 2016.

[238] J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum. “KORE: keyphrase overlap relatedness for entity disambiguation”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2012.

[239] J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, and G. Weikum. “ YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia ”. In: *The Web Conference (WWW)*. 2011.

[240] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. “YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia”. *Artificial Intelligence (AI)*. 194: 28–61. 2013.

[241] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. “Robust Disambiguation of Named Entities in Text”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2011.

[242] R. Hoffmann, C. Zhang, and D. S. Weld. “Learning 5000 Relational Extractors”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2010.

[243] T. Hofmann. “Unsupervised Learning by Probabilistic Latent Semantic Analysis”. *Machine Learning*. 42(1/2): 177–196. 2001.

[244] A. Hogan. *The Web of Data*. Springer, 2020.

[245] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A. N. Ngomo, S. M. Rashid, A. Rula, L. Schmelzeisen, J. F. Sequeda, S. Staab, and A. Zimmermann. "Knowledge Graphs". *arXiv*. abs/2003.02320. 2020.

[246] A. Hopkinson, A. Gurdasani, D. Palfrey, and A. Mittal. "Demand-Weighted Completeness Prediction for a Knowledge Base". In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[247] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. "Information Retrieval in Folksonomies: Search and Ranking". In: *European Semantic Web Conference (ESWC)*. 2006.

[248] M. B. Hoy. "Wolfram|Alpha: a brief introduction". *Medical reference services quarterly*. 29(1): 67–74. 2010.

[249] Y. Hua, C. Danescu-Niculescu-Mizil, D. Taraborelli, N. Thain, J. Sorensen, and L. Dixon. "WikiConv: A Corpus of the Complete Conversational History of a Large Online Collaborative Community". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.

[250] Z. Huang, W. Xu, and K. Yu. "Bidirectional LSTM-CRF Models for Sequence Tagging". *CoRR abs/1508.01991*. 2015.

[251] T. Hubauer, S. Lamparter, P. Haase, and D. M. Herzig. "Use Cases of the Industrial Knowledge Graph at Siemens". In: *International Semantic Web Conference (ISWC)*. 2018.

[252] J. M. van Hulst, F. Hasibi, K. Dercksen, K. Balog, and A. P. de Vries. "REL: An Entity Linker Standing on the Shoulders of Giants". In: *ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 2020.

[253] Y. Ibrahim, M. Riedewald, and G. Weikum. "Making Sense of Entities and Quantities in Web Tables". In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2016.

[254] Y. Ibrahim, M. Riedewald, G. Weikum, and D. Zeinalipour-Yazti. "Bridging Quantities in Tables and Text". In: *IEEE International Conference on Data Engineering (ICDE)*. 2019.

[255] I. F. Ilyas and X. Chu. *Data Cleaning*. ACM, 2019.

[256] I. F. Ilyas and X. Chu. "Trends in Cleaning Relational Data: Consistency and Deduplication". *Foundations and Trends in Databases*. 5(4): 281–393. 2015.

[257] I. F. Ilyas, V. Markl, P. J. Haas, P. Brown, and A. Aboulnaga. "CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies". In: *ACM Conference on Management of Data (SIGMOD)*. 2004.

[258] P. G. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano. "Towards a query optimizer for text-centric tasks". *ACM Transactions on Database Systems (TODS)*. 32(4): 21. 2007.

[259] A. Ismayilov, D. Kontokostas, S. Auer, J. Lehmann, and S. Hellmann. "Wikidata through the eyes of DBpedia". *Semantic Web Journal (SWJ)*. 9(4): 493–503. 2018.

[260] A. Jain, P. G. Ipeirotis, and L. Gravano. "Building query optimizers for information extraction: the SQoUT project". *SIGMOD Record*. 37(4): 28–34. 2008.

[261] P. Jansson and S. Liu. "Topic modelling enriched LSTM models for the detection of novel and emerging named entities from social media". In: *International Conference on Big Data*. 2017.

[262] R. Jäschke, A. Hotho, C. Schmitz, B. Ganter, and G. Stumme. "Discovering shared conceptualizations in folksonomies". *Journal of Web Semantics (JWS)*. 6(1): 38–53. 2008.

[263] G. Jeh and J. Widom. "Scaling personalized web search". In: *The Web Conference (WWW)*. 2003.

[264] G. Ji, K. Liu, S. He, and J. Zhao. "Distant Supervision for Relation Extraction with Sentence-Level Attention and Entity Descriptions". In: *Conference on Artificial Intelligence (AAAI)*. 2017.

[265] H. Ji, R. Grishman, H. T. Dang, K. Griffitt, and J. Ellis. "Overview of the TAC 2010 knowledge base population track". In: *Text Analysis Conference (TAC)*. 2010.

[266] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu. "A Survey on Knowledge Graphs: Representation, Acquisition and Applications". *CoRR*. abs/2002.00388. 2020.

[267] Z. Ji, Q. Wei, and H. Xu. "BERT-based Ranking for Biomedical Entity Normalization". *CoRR*. abs/1908.03548. 2019.

[268] S. Jiang, S. Baumgartner, A. Ittycheriah, and C. Yu. "Factoring Fact-Checks: Structured Information Extraction from Fact-Checking Articles". In: *The Web Conference (WWW)*. 2020. 1592–1603.

[269] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig. "How Can We Know What Language Models Know". *Transactions of the Association for Computational Linguistics (TACL)*. 8: 423–438. 2020.

[270] Jimmy, G. Zuccon, and B. Koopman. "Choices in Knowledge-Base Retrieval for Consumer Health Search". In: *European Conference on Information Retrieval (ECIR)*. 2018.

[271] P. Jindal and D. Roth. "Extraction of events and temporal expressions from clinical narratives". *Journal of Biomedical Informatics*. 46(6): S13–S19. 2013.

[272] M. Joshi, O. Levy, L. Zettlemoyer, and D. S. Weld. "BERT for Coreference Resolution: Baselines and Analysis". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019.

[273] D. Jurafsky and J. H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 3rd Edition*. Prentice Hall, Pearson Education International, 2019.

[274] V. Kalokyri, A. Borgida, and A. Marian. "YourDigitalSelf: A Personal Digital Trace Integration Tool". In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2018.

[275] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. "The Eigentrust algorithm for reputation management in P2P networks". In: *The Web Conference (WWW)*. 2003.

[276] N. Karalis, G. M. Mandilaras, and M. Koubarakis. "Extending the YAGO2 Knowledge Graph with Precise Geospatial Knowledge". In: *International Semantic Web Conference (ISWC)*. 2019. 181–197.

[277] K. Kari. "Building, and communicating, a knowledge graph in Zalando". In: *Keynote at DINAcon (Konferenz für Digitale Nachhaltigkeit), https://www.youtube.com/watch?v=QkgAFKL26Vg and https://dinacon.ch/en/interview-with-katariina-kari/*. 2019.

[278] G. Kasneci, M. Ramanath, F. M. Suchanek, and G. Weikum. "The YAGO-NAGA approach to knowledge discovery". *SIGMOD Record*. 37(4): 41–47. 2008.

[279] M. Kejriwal and P. Szekely. "Knowledge graphs for social good: an entity-centric search engine for the human trafficking domain". *IEEE Transactions on Big Data*. 2017.

[280] K. A. Khatib, Y. Hou, H. Wachsmuth, C. Jochim, F. Bonin, and B. Stein. "End-to-End Argumentation Knowledge Graph Construction". In: *Conference on Artificial Intelligence (AAAI)*. 2020.

[281] H. Knublauch and D. Kontokostas. "Shapes Constraint Language (SHACL), https://www.w3.org/TR/2017/REC-shacl-20170720/". *W3C Recommendation*. 2017.

[282] N. Kolitsas, O. Ganea, and T. Hofmann. "End-to-End Neural Entity Linking". In: *Conference on Computational Natural Language Learning (CoNLL)*. 2018.

[283] D. Koller and N. Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[284] P. Konda, S. Das, P. S. G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. F. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra. "Magellan: Toward Building Entity Matching Management Systems". *Proceedings of the VLDB Endowment*. 9(12): 1197–1208. 2016.

[285] D. Kontokostas, C. Bratsas, S. Auer, S. Hellmann, I. Antoniou, and G. Metakides. "Internationalization of Linked Data: The case of the Greek DBpedia edition". *Journal of Web Semantics (JWS)*. 15: 51–61. 2012.

[286] B. Koopman and G. Zuccon. "WSDM 2019 Tutorial on Health Search (HS2019): A Full-Day from Consumers to Clinicians (with materials on https://github.com/ielab/health-search-tutorial/tree/wsdm2019)". In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2019. 838–839.

[287] H. Köpcke and E. Rahm. "Frameworks for entity matching: A comparison". *Data & Knowledge Engineering*. 69(2): 197–210. 2010.

[288] Y. Koren and R. M. Bell. "Advances in Collaborative Filtering". In: *Recommender Systems Handbook*. 2015. 77–118.

[289] Y. Koren, R. M. Bell, and C. Volinsky. "Matrix Factorization Techniques for Recommender Systems". *IEEE Computer*. 42(8): 30–37. 2009.

[290] P. Kouki, J. Pujara, C. Marcum, L. M. Koehly, and L. Getoor. "Collective Entity Resolution in Familial Networks". In: *IEEE Conference on Data Mining (ICDM)*. 2017.

[291] P. Kouki, J. Pujara, C. Marcum, L. M. Koehly, and L. Getoor. "Collective entity resolution in multi-relational familial networks". *Knowledge and Information Systems (KAIS)*. 61(3): 1547–1581. 2019.

[292] Z. Kozareva and E. H. Hovy. "Learning Arguments and Supertypes of Semantic Relations Using Recursive Patterns". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2010.

[293] S. Krause, L. Hennig, A. Moro, D. Weissenborn, F. Xu, H. Uszkoreit, and R. Navigli. "Sar-graphs: A language resource connecting linguistic knowledge with semantic relations from knowledge graphs". *Journal of Web Semantics (JWS)*. 37-38: 112–131. 2016.

[294] S. Krause, H. Li, H. Uszkoreit, and F. Xu. "Large-Scale Learning of Relation-Extraction Rules with Distant Supervision from the Web". In: *International Semantic Web Conference (ISWC)*. 2012.

[295] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations". *International Journal of Computer Vision (IJCV)*. 123(1): 32–73. 2017.

[296] M. Krötzsch, D. Vrandecic, and M. Völkel. "Semantic MediaWiki". In: *International Semantic Web Conference (ISWC)*. 2006.

[297] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. "Collective annotation of Wikipedia entities in web text". In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2009.

[298] S. Kumar, R. West, and J. Leskovec. "Disinformation on the Web: Impact, Characteristics, and Detection of Wikipedia Hoaxes". In: *The Web Conference (WWW)*. 2016.

[299] N. Kushmerick. "Wrapper induction: Efficiency and expressiveness". *Artificial Intelligence (AI)*. 118(1-2): 15–68. 2000.

[300] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. “Wrapper Induction for Information Extraction”. In: *Joint Conference on Artificial Intelligence (IJCAI)*. 1997.

[301] E. Kuzey, V. Setty, J. Strötgen, and G. Weikum. “As Time Goes By: Comprehensive Tagging of Textual Phrases with Temporal Scopes”. In: *The Web Conference (WWW)*. 2016.

[302] E. Kuzey, J. Vreeken, and G. Weikum. “A Fresh Look on Knowledge Bases: Distilling Named Events from News”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2014.

[303] E. Kuzey and G. Weikum. “Extraction of temporal facts and events from Wikipedia”. In: *Temporal Web Analytics Workshop, co-located with the Web Conference*. 2012.

[304] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *International Conference on Machine Learning (ICML)*. 2001.

[305] J. Lajus, L. Galárraga, and F. M. Suchanek. “Fast and Exact Rule Mining with AMIE 3”. In: *European Semantic Web Conference (ESWC)*. 2020.

[306] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. “Neural Architectures for Named Entity Recognition”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2016.

[307] W. Lang, R. V. Nehme, E. Robinson, and J. F. Naughton. “Partial results in database systems”. In: *ACM Conference on Management of Data (SIGMOD)*. 2014.

[308] N. Lao, E. Minkov, and W. W. Cohen. “Learning Relational Features with Backward Random Walks”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2015.

[309] N. Lao, T. M. Mitchell, and W. W. Cohen. “Random Walk Inference and Learning in A Large Scale Knowledge Base”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2011.

[310] N. Lao, T. Mitchell, and W. W. Cohen. “Random walk inference and learning in a large scale knowledge base”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2011.

[311] N. Lao, A. Subramanya, F. Pereira, and W. W. Cohen. “Reading the web with learned syntactic-semantic inference rules”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2012.

[312] N. Lazic, A. Subramanya, M. Ringgaard, and F. Pereira. “Plato: A Selective Context Model for Entity Resolution”. *Transactions of the Association for Computational Linguistics (TACL)*. 3: 503–515. 2015.

[313] H. Lee, A. X. Chang, Y. Peirsman, N. Chambers, M. Surdeanu, and D. Jurafsky. "Deterministic Coreference Resolution Based on Entity-Centric, Precision-Ranked Rules". *Computational Linguistics*. 39(4): 885–916. 2013.

[314] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". *Bioinformatics*. 36(4): 1234–1240. 2019. ISSN: 1367-4803.

[315] K. Lee, L. He, M. Lewis, and L. Zettlemoyer. "End-to-end Neural Coreference Resolution". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017.

[316] K. Leetaru and P. A. Schrodt. "Gdelt: Global data on events, location, and tone, 1979–2012". In: *ISA Annual Convention*. 2013. 1–49.

[317] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. "DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia". *Semantic Web Journal (SWJ)*. 6(2): 167–195. 2015.

[318] O. Lehmberg and C. Bizer. "Stitching Web Tables for Improving Matching Quality". *Proceedings of the VLDB Endowment*. 10(11): 1502–1513. 2017.

[319] O. Lehmberg and C. Bizer. "Synthesizing N-ary Relations from Web Tables". In: *International Conference on Web Intelligence, Mining and Semantics*. 2019.

[320] C. Lei, F. özcan, A. Quamar, A. R. Mittal, J. Sen, D. Saha, and K. Sankaranarayanan. "Ontology-Based Natural Language Query Interfaces for Data Exploration". *IEEE Data Engineering Bulletin*. 41(3): 52–63. 2018.

[321] J. L. Leidner. *Toponym resolution in text: Annotation, evaluation and applications of spatial grounding of place names*. Universal-Publishers, 2008.

[322] D. B. Lenat. "CYC: A Large-Scale Investment in Knowledge Infrastructure". *Communication of the ACM*. 38(11): 32–38. 1995.

[323] D. B. Lenat and E. A. Feigenbaum. "On the Thresholds of Knowledge". *Artificial Intelligence (AI)*. 47(1-3): 185–250. 1991.

[324] O. Levy, Y. Goldberg, and I. Dagan. "Improving Distributional Similarity with Lessons Learned from Word Embeddings". *Transactions of the Association for Computational Linguistics (TACL)*. 3: 211–225. 2015.

[325] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer. "Zero-Shot Relation Extraction via Reading Comprehension". In: *Conference on Computational Natural Language Learning (CoNLL)*. 2017.

[326] C. M. Li and F. Manyà. "MaxSAT, Hard and Soft Constraints". In: *Handbook of Satisfiability*. IOS Press, 2009. 613–631.

[327] F. Li, X. L. Dong, A. Langen, and Y. Li. "Knowledge Verification for LongTail Verticals". *Proceedings of the VLDB Endowment*. 10(11): 1370–1381. 2017.

[328] G. Li. "Special Issue on Large-Scale Data Integration". *IEEE Data Engineering Bulletin*. 41(2): 2. 2018.

[329] H. Li. *Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition*. Morgan & Claypool Publishers, 2014.

[330] J. Li, A. Sun, J. Han, and C. Li. "A Survey on Deep Learning for Named Entity Recognition". *CoRR*. abs/1812.09449. 2018.

[331] J. Li, J. Tang, Y. Li, and Q. Luo. "RiMOM: A Dynamic Multistrategy Ontology Alignment Framework". *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 21(8): 1218–1232. 2009.

[332] Q. Li and H. Ji. "Incremental Joint Extraction of Entity Mentions and Relations". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2014.

[333] X. Li, W. Meng, and C. T. Yu. "T-verifier: Verifying truthfulness of fact statements". In: *IEEE International Conference on Data Engineering (ICDE)*. 2011.

[334] X. Li, F. Yin, Z. Sun, X. Li, A. Yuan, D. Chai, M. Zhou, and J. Li. "Entity-Relation Extraction as Multi-Turn Question Answering". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019.

[335] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han. "A Survey on Truth Discovery". *SIGKDD Explorations*. 17(2): 1–16. 2015.

[336] Y. Li, S. Tan, H. Sun, J. Han, D. Roth, and X. Yan. "Entity Disambiguation with Linkless Knowledge Bases". In: *The Web Conference (WWW)*. 2016.

[337] M. D. Lieberman and J. J. Lin. "You Are Where You Edit: Locating Wikipedia Contributors through Edit Histories". In: *International Conference on Weblogs and Social Media (ICWSM)*. 2009.

[338] G. Limaye, S. Sarawagi, and S. Chakrabarti. "Annotating and Searching Web Tables Using Entities, Types and Relationships". *Proceedings of the VLDB Endowment*. 3(1): 1338–1347. 2010.

[339] D. Lin and P. Pantel. "DIRT – Discovery of inference rules from text". In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2001.

[340] X. Lin, H. Li, H. Xin, Z. Li, and L. Chen. "KBPearl: A Knowledge Base Population System Supported by Joint Entity and Relation Linking". *Proceedings of the VLDB Endowment*. 13(7): 1035–1049. 2020.

[341] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. "Learning Entity and Relation Embeddings for Knowledge Graph Completion". In: *Conference on Artificial Intelligence (AAAI)*. 2015.

[342] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun. "Neural Relation Extraction with Selective Attention over Instances". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2016.

[343] X. Ling, S. Singh, and D. S. Weld. “Design Challenges for Entity Linking”. *Transactions of the Association for Computational Linguistics (TACL)*. 3: 315–328. 2015.

[344] X. Ling and D. S. Weld. “Fine-Grained Entity Recognition”. In: *Conference on Artificial Intelligence (AAAI)*. 2012.

[345] X. Ling and D. S. Weld. “Temporal Information Extraction”. In: *Conference on Artificial Intelligence (AAAI)*. 2010.

[346] B. Liu, W. Guo, D. Niu, J. Luo, C. Wang, Z. Wen, and Y. Xu. “GIANT: Scalable Creation of a Web-scale Ontology”. In: *ACM Conference on Management of Data (SIGMOD)*. ACM, 2020.

[347] B. Liu, W. Guo, D. Niu, C. Wang, S. Xu, J. Lin, K. Lai, and Y. Xu. “A User-Centered Concept Mining System for Query and Document Understanding at Tencent”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2019.

[348] G. Liu, X. Li, J. Wang, M. Sun, and P. Li. “Extracting Knowledge from Web Text with Monte Carlo Tree Search”. In: *The Web Conference (WWW)*. 2020.

[349] H. Liu, L. Hunter, V. Keselj, and K. Verspoor. “Approximate subgraph matching-based literature mining for biomedical events and relations”. *PloS one*. 8(4). 2013.

[350] T. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.

[351] X. Liu, Y. Li, H. Wu, M. Zhou, F. Wei, and Y. Lu. “Entity Linking for Tweets”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2013.

[352] Y. Liu, F. Wei, S. Li, H. Ji, M. Zhou, and H. Wang. “A Dependency-Based Neural Network for Relation Classification”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2015.

[353] K. Lo, L. L. Wang, M. Neumann, R. Kinney, and D. S. Weld. “S2ORC: The Semantic Scholar Open Research Corpus”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020.

[354] C. Lockard, X. L. Dong, P. Shiralkar, and A. Einolghozati. “CERES: Distantly Supervised Relation Extraction from the Semi-Structured Web”. *Proceedings of the VLDB Endowment*. 11(10): 1084–1096. 2018.

[355] C. Lockard, P. Shiralkar, and X. L. Dong. “OpenCeres: When Open Information Extraction Meets the Semi-Structured Web”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2019.

[356] C. Lockard, P. Shiralkar, X. L. Dong, and H. Hajishizi. “ZeroShotCeres: Zero-shot relation extraction from semi-structured webpages”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020.

[357] L. Logeswaran, M. Chang, K. Lee, K. Toutanova, J. Devlin, and H. Lee. "Zero-Shot Entity Linking by Reading Entity Descriptions". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019.

[358] Y. Luan, L. He, M. Ostendorf, and H. Hajishirzi. "Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.

[359] M. Luggen, D. Difallah, C. Sarasua, G. Demartini, and P. Cudré-Mauroux. "Non-Parametric Class Completeness Estimators for Collaborative Knowledge Graphs - The Case of Wikidata". In: *International Semantic Web Conference (ISWC)*. 2019.

[360] X. Luo, L. Liu, Y. Yang, L. Bo, Y. Cao, J. Wu, Q. Li, K. Yang, and K. Q. Zhu. "AliCoCo: Alibaba E-commerce Cognitive Concept Net". In: *ACM Conference on Management of Data (SIGMOD)*. ACM, 2020.

[361] X. Ma and E. H. Hovy. "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2016.

[362] A. Madaan, A. Mittal, Mausam, G. Ramakrishnan, and S. Sarawagi. "Numerical Relation Extraction with Minimal Supervision". In: *Conference on Artificial Intelligence (AAAI)*. 2016.

[363] A. Maedche and S. Staab. "Discovering Conceptual Relations from Text". In: *European Conference on Artificial Intelligence (ECAI)*. 2000.

[364] F. Mahdisoltani, J. A. Biega, and F. M. Suchanek. " YAGO3: A Knowledge Base from Multilingual Wikipedias ". In: *Conference on Innovative Data Systems Research (CIDR)*. 2015.

[365] K. Mai, T. Pham, M. T. Nguyen, N. T. Duc, D. Bollegala, R. Sasano, and S. Sekine. "An Empirical Study on Fine-Grained Named Entity Recognition". In: *Conference on Computational Linguistics (COLING)*. 2018.

[366] S. Malyshev, M. Krötzsch, L. González, J. Gonsior, and A. Bielefeldt. "Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph". In: *International Semantic Web Conference (ISWC)*. 2018.

[367] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2014.

[368] Y. Mao, X. Ren, J. Shen, X. Gu, and J. Han. "End-to-End Reinforcement Learning for Automatic Taxonomy Induction". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2018. 2462–2472.

[369] J. Marin, A. Biswas, F. Ofli, N. Hynes, A. Salvador, Y. Aytar, I. Weber, and A. Torralba. "Recipe1M: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images". *CoRR*. abs/1810.06553. 2018.

[370] J. Martinez-Rodriguez, A. Hogan, and I. Lopez-Arvalo. "Information extraction meets the Semantic Web: A survey". *Semantic Web Journal (SWJ)*. 11(2): 255–335. 2020.

[371] G. D. S. Martino, S. Cresci, A. Barron-Cedeño, S. Yu, R. D. Pietro, and P. Nakov. "A Survey on Computational Propaganda Detection". In: *Joint Conference on Artificial Intelligence (IJCAI)*. 2020. 4826–4832.

[372] C. Matuszek, M. J. Witbrock, R. C. Kahlert, J. Cabral, D. Schneider, P. Shah, and D. B. Lenat. "Searching for Common Sense: Populating Cyc from the Web". In: *Conference on Artificial Intelligence (AAAI)*. 2005.

[373] Mausam. "Open Information Extraction Systems and Downstream Applications". In: *Joint Conference on Artificial Intelligence (IJCAI)*. 2016.

[374] Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni. "Open Language Learning for Information Extraction". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2012.

[375] E. Meij. "Understanding News Using the Bloomberg Knowledge Graph". In: *Talk at Big Data Innovators Gathering (BIG) at the Web Conference 2019, https://speakerdeck.com/emeij/understanding-news-using-the-bloomberg-knowledge-graph*.

[376] C. Meilicke, M. W. Chekol, D. Ruffinelli, and H. Stuckenschmidt. "Anytime Bottom-Up Rule Learning for Knowledge Graph Completion". In: *Joint Conference on Artificial Intelligence (IJCAI)*. 2019.

[377] F. Menczer, G. Pant, and P. Srinivasan. "Topical web crawlers: Evaluating adaptive algorithms". *ACM Transactions on Internet Technology (TOIT)*. 4(4): 378–419. 2004.

[378] P. N. Mendes, M. Jakob, and C. Bizer. "DBpedia: A Multilingual Cross-domain Knowledge Base". In: *Conference on Language Resources and Evaluation (LREC)*. 2012.

[379] P. N. Mendes, M. Jakob, A. Garcia-Silva, and C. Bizer. "DBpedia spotlight: shedding light on the web of documents". In: *International Conference on Semantic Systems*. 2011.

[380] F. Mesquita, M. Cannaviccio, J. Schmidek, P. Mirza, and D. Barbosa. "KnowledgeNet: A Benchmark Dataset for Knowledge Base Population". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019.

[381] R. Meusel, P. Petrovski, and C. Bizer. "The WebDataCommons Microdata, RDFa and Microformat Dataset Series". In: *International Semantic Web Conference (ISWC)*. 2014.

[382] J. Michael, G. Stanovsky, L. He, I. Dagan, and L. Zettlemoyer. “Crowdsourcing Question-Answer Meaning Representations”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[383] R. Mihalcea and A. Csomai. “Wikify!: linking documents to encyclopedic knowledge”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2007.

[384] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. “Distributed Representations of Words and Phrases and their Compositionality”. In: *Neural Information Processing Systems (NeurIPS)*. 2013.

[385] R. J. Miller, F. Nargesian, E. Zhu, C. Christodoulakis, K. Q. Pu, and P. Andritsos. “Making Open Data Transparent: Data Discovery on Open Data”. *IEEE Data Engineering Bulletin*. 41(2): 59–70. 2018.

[386] D. N. Milne and I. H. Witten. “Learning to link with wikipedia”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2008.

[387] B. Min, S. Shi, R. Grishman, and C. Lin. “Ensemble Semantics for Large-scale Unsupervised Relation Extraction”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2012.

[388] B. Minaei-Bidgoli, R. Barmaki, and M. Nasiri. “Mining numerical association rules via multi-objective genetic algorithms”. *Information Sciences*. 2013.

[389] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. “Distant supervision for relation extraction without labeled data”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2009.

[390] P. Mirza, S. Razniewski, F. Darari, and G. Weikum. “Cardinal Virtues: Extracting Relation Cardinalities from Text”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2017.

[391] P. Mirza, S. Razniewski, F. Darari, and G. Weikum. “Enriching Knowledge Bases with Counting Quantifiers”. In: *International Semantic Web Conference (ISWC)*. 2018.

[392] P. Mirza and S. Tonelli. “CATENA: CAusal and TEmporal relation extraction from NAtural language texts”. In: *Conference on Computational Linguistics (COLING)*. 2016.

[393] B. D. Mishra, N. Tandon, and P. Clark. “Domain-Targeted, High Precision Knowledge Extraction”. *Transactions of the Association for Computational Linguistics (TACL)*. 5: 233–246. 2017.

[394] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. “Never-Ending Learning”. In: *Conference on Artificial Intelligence (AAAI)*. 2015.

[395] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. D. Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. A. Platanios, A. Ritter, M. Samadi, B. Settles, R. C. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. "Never-ending learning". *Communication of the ACM*. 61(5): 103–115. 2018.

[396] M. Miwa and M. Bansal. "End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2016.

[397] T. Mohamed, E. R. H. Jr., and T. M. Mitchell. "Discovering Relations between Noun Categories". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2011.

[398] D. Montoya, T. P. Tanon, S. Abiteboul, P. Senellart, and F. M. Suchanek. "A Knowledge Base for Personal Information Management". In: *Workshop on Linked Data on the Web (LDOW), co-located with the Web Conference*. 2018.

[399] A. Moro, A. Raganato, and R. Navigli. "Entity Linking meets Word Sense Disambiguation: a Unified Approach". *Transactions of the Association for Computational Linguistics (TACL)*. 2: 231–244. 2014.

[400] A. Moschitti. "A Study on Convolution Kernels for Shallow Statistic Parsing". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2004.

[401] A. Moschitti. "Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees". In: *European Conference on Machine Learning (ECML)*. 2006.

[402] A. Moschitti, D. Pighin, and R. Basili. "Semantic Role Labeling via Tree Kernel Joint Inference". In: *Conference on Computational Natural Language Learning (CoNLL)*. 2006.

[403] A. Motro. "Integrity= validity+ completeness". *ACM Transactions on Database Systems (TODS)*. 14(4): 480–502. 1989.

[404] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[405] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. "Deep Learning for Entity Matching: A Design Space Exploration". In: *ACM Conference on Management of Data (SIGMOD)*. 2018.

[406] D. Mueller and G. Durrett. "Effective Use of Context in Noisy Entity Linking". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.

[407] S. Muggleton and L. D. Raedt. "Inductive Logic Programming: Theory and Methods". *Journal of Logic Programming*. 19/20: 629–679. 1994.

[408] S. Mukherjee, G. Weikum, and C. Danescu-Niculescu-Mizil. “People on drugs: credibility of user statements in health communities”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2014.

[409] J. W. Murdock, A. Kalyanpur, C. Welty, J. Fan, D. A. Ferrucci, D. Gondek, L. Zhang, and H. Kanayama. “Typing candidate answers using type coercion”. *IBM Journal of Research and Development*. 56(3): 7. 2012.

[410] I. Muslea, S. Minton, and C. A. Knoblock. “Hierarchical Wrapper Induction for Semistructured Information Sources”. *Autonomous Agents and Multi-Agent Systems*. 4(1/2): 93–114. 2001.

[411] N. Nakashole and T. M. Mitchell. “Language-Aware Truth Assessment of Fact Candidates”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2014.

[412] N. Nakashole, M. Theobald, and G. Weikum. “Scalable knowledge harvesting with high precision and high recall”. In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2011.

[413] N. Nakashole, T. Tylenda, and G. Weikum. “Fine-grained Semantic Typing of Emerging Entities”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2013.

[414] N. Nakashole, G. Weikum, and F. M. Suchanek. “PATTY: A Taxonomy of Relational Patterns with Semantic Types”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2012.

[415] V. Nastase and M. Strube. “Transforming Wikipedia into a large scale multilingual concept network”. *Artificial Intelligence (AI)*. 194: 62–85. 2013.

[416] F. Naumann and M. Herschel. *An Introduction to Duplicate Detection*. Morgan & Claypool Publishers, 2010.

[417] R. Navigli. “Word sense disambiguation: A survey”. *ACM Computing Surveys*. 41(2): 10:1–10:69. 2009.

[418] R. Navigli and S. P. Ponzetto. “BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network”. *Artificial Intelligence (AI)*. 193: 217–250. 2012.

[419] M. Nentwig, M. Hartung, A. N. Ngomo, and E. Rahm. “A survey of current Link Discovery frameworks”. *Semantic Web Journal (SWJ)*. 8(3): 419–436. 2017.

[420] D. B. Nguyen, A. Abujabal, K. Tran, M. Theobald, and G. Weikum. “Query-Driven On-The-Fly Knowledge Base Construction”. *Proceedings of the VLDB Endowment*. 11(1): 66–79. 2017.

[421] D. B. Nguyen, J. Hoffart, M. Theobald, and G. Weikum. “AIDA-light: High-Throughput Named-Entity Disambiguation”. In: *Workshop on Linked Data on the Web (LDOW), co-located with the Web Conference*. 2014.

[422] D. B. Nguyen, M. Theobald, and G. Weikum. “J-NERD: Joint Named Entity Recognition and Disambiguation with Rich Linguistic Features”. *Transactions of the Association for Computational Linguistics (TACL)*. 4: 215–229. 2016.

[423] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. “A Review of Relational Machine Learning for Knowledge Graphs”. *Proceedings of the IEEE*. 104(1): 11–33. 2016.

[424] M. Nickel, V. Tresp, and H. Kriegel. “Factorizing YAGO: scalable machine learning for linked data”. In: *The Web Conference (WWW)*. 2012.

[425] Z. Nie, J. Wen, and W. Ma. “Statistical Entity Extraction From the Web”. *Proceedings of the IEEE*. 100(9): 2675–2687. 2012.

[426] F. Å. Nielsen, D. Mietchen, and E. L. Willighagen. “Scholia, Scientometrics and Wikidata”. In: *Scientometrics Workshop*. 2017.

[427] Q. Ning, S. Subramanian, and D. Roth. “An Improved Neural Baseline for Temporal Relation Extraction”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019.

[428] F. Niu, C. Ré, A. Doan, and J. W. Shavlik. “Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS”. *Proceedings of the VLDB Endowment*. 4(6): 373–384. 2011.

[429] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor. “Industry-Scale Knowledge Graphs: Lessons and Challenges”. *Communication of the ACM*. 62(8): 36–43. 2019.

[430] A. Oelen, M. Y. Jaradeh, M. Stocker, and S. Auer. “Generate FAIR Literature Surveys with Scholarly Knowledge Graphs”. In: *Joint Conference on Digital Libraries*. 2020.

[431] A. Olteanu, C. Castillo, F. Diaz, and E. Kiciman. “Social Data: Biases, Methodological Pitfalls, and Ethical Boundaries”. *Frontiers Big Data*. 2: 13. 2019.

[432] P. E. O’Neil and E. J. O’Neil. *Database: Principles, Programming, and Performance, Second Edition*. Morgan Kaufmann, 2000.

[433] S. Ortona, V. V. Meduri, and P. Papotti. “Robust Discovery of Positive and Negative Rules in Knowledge Bases”. In: *IEEE International Conference on Data Engineering (ICDE)*. 2018.

[434] S. Ortona, V. V. Meduri, and P. Papotti. “RuDiK: Rule Discovery in Knowledge Bases”. *Proceedings of the VLDB Endowment*. 11(12): 1946–1949. 2018.

[435] Y. Oulabi and C. Bizer. “Extending Cross-Domain Knowledge Bases with Long Tail Entities using Web Table Data”. In: *Conference on Extending Database Technology (EDBT)*. 2019.

[436] H. Pal and Mausam. “Demonyms and Compound Relational Nouns in Nominal Open IE”. In: *Workshop on Automated Knowledge Base Construction (AKBC)*. 2016.

[437] K. Pal, V. T. Ho, and G. Weikum. "Co-Clustering Triples from Open Information Extraction". In: *CoDS-COMAD Conference*. 2020.
[438] M. Palmer, D. Gildea, and N. Xue. *Semantic Role Labeling*. Morgan & Claypool Publishers, 2010.
[439] M. Palmer, P. R. Kingsbury, and D. Gildea. "The Proposition Bank: An Annotated Corpus of Semantic Roles". *Computational Linguistics*. 31(1): 71–106. 2005.
[440] M. Pasca. "Finding Needles in an Encyclopedic Haystack: Detecting Classes Among Wikipedia Articles". In: *The Web Conference (WWW)*. 2018.
[441] M. Pasca. "Open-Domain Fine-Grained Class Extraction from Web Search Queries". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2013.
[442] M. Pasca. "The Role of Query Sessions in Interpreting Compound Noun Phrases". In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2015.
[443] M. Pasca and B. V. Durme. "What You Seek Is What You Get: Extraction of Class Attributes from Query Logs". In: *Joint Conference on Artificial Intelligence (IJCAI)*. 2007.
[444] H. Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods". *Semantic Web Journal (SWJ)*. 2017.
[445] E. Pavlick, P. Rastogi, J. Ganitkevitch, B. V. Durme, and C. Callison-Burch. "PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2015.
[446] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W. Yih. "Cross-Sentence N-ary Relation Extraction with Graph LSTMs". *Transactions of the Association for Computational Linguistics (TACL)*. 5: 101–115. 2017.
[447] J. Pennington, R. Socher, and C. D. Manning. "Glove: Global Vectors for Word Representation". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.
[448] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. "Deep Contextualized Word Representations". In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.
[449] F. Petroni, T. Rocktäschel, S. Riedel, P. S. H. Lewis, A. Bakhtin, Y. Wu, and A. H. Miller. "Language Models as Knowledge Bases?" In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019.
[450] A. Pfadler, H. Zhao, J. Wang, L. Wang, P. Huang, and D. L. Lee. "Billion-scale Recommendation with Heterogeneous Side Information at Taobao". In: *IEEE International Conference on Data Engineering (ICDE)*. 2020.

[451] F. Piccinno and P. Ferragina. “From TagME to WAT: a new entity annotator”. In: *International Workshop on Entity Recognition & Disambiguation, co-located with ACM SIGIR Conference*. 2014.

[452] A. Piscopo, L. Kaffee, C. Phethean, and E. Simperl. “Provenance Information in a Collaborative Knowledge Graph: An Evaluation of Wikidata External References”. In: *International Semantic Web Conference (ISWC)*. 2017.

[453] A. Piscopo, C. Phethean, and E. Simperl. “What Makes a Good Collaborative Knowledge Graph: Group Composition and Quality in Wikidata”. In: *International Conference on Social Informatics*. 2017.

[454] A. Piscopo and E. Simperl. “What we talk about when we talk about wikidata quality: a literature survey”. In: *International Symposium on Open Collaboration*. 2019.

[455] M. Ponza, P. Ferragina, and S. Chakrabarti. “On Computing Entity Relatedness in Wikipedia, with Applications”. *Knowledge Based Systems*. 188. 2020.

[456] S. P. Ponzetto and M. Strube. “Deriving a Large-Scale Taxonomy from Wikipedia”. In: *Conference on Artificial Intelligence (AAAI)*. 2007.

[457] S. P. Ponzetto and M. Strube. “Taxonomy induction based on a collaboratively built knowledge repository”. *Artificial Intelligence (AI)*. 175(9-10): 1737–1756. 2011.

[458] H. Poon and P. M. Domingos. “Unsupervised Ontology Induction from Text”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2010.

[459] K. Popat, S. Mukherjee, J. Strötgen, and G. Weikum. “Where the Truth Lies: Explaining the Credibility of Emerging Claims on the Web and Social Media”. In: *The Web Conference (WWW)*. 2017. 1003–1012.

[460] S. S. Pradhan, E. H. Hovy, M. P. Marcus, M. Palmer, L. A. Ramshaw, and R. M. Weischedel. “Ontonotes: a Unified Relational Semantic Representation”. *International Journal of Semantic Computing*. 1(4): 405–419. 2007.

[461] R. E. Prasojo, M. Kacimi, and W. Nutt. “StuffIE: Semantic Tagging of Unlabeled Facets Using Fine-Grained Information Extraction”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2018.

[462] N. Prokoshyna, J. Szlichta, F. Chiang, R. J. Miller, and D. Srivastava. “Combining Quantitative and Logical Data Cleaning”. *Proceedings of the VLDB Endowment*. 9(4): 300–311. 2015.

[463] J. Pujara, H. Miao, L. Getoor, and W. W. Cohen. “Knowledge Graph Identification”. In: *International Semantic Web Conference (ISWC)*. 2013.

[464] J. Pujara, H. Miao, L. Getoor, and W. W. Cohen. “Large-Scale Knowledge Graph Identification Using PSL”. In: *AAAI Fall Symposia*. 2013.

[465] V. Punyakanok, D. Roth, and W. Yih. “The Importance of Syntactic Parsing and Inference in Semantic Role Labeling”. *Computational Linguistics*. 34(2): 257–287. 2008.

[466] Y. Qi and J. Xiao. “Fintech: AI powers financial services to improve people’s lives”. *Communication of the ACM*. 61(11): 65–69. 2018.

[467] R. Qiang. “Understand Your World with Bing”. In: *Bing Blog, 21 March 2013, https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/*.

[468] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang. “Pre-trained Models for Natural Language Processing: A Survey”. *CoRR*. abs/2003.08271. 2020.

[469] M. Qu, X. Ren, and J. Han. “Automatic Synonym Discovery with Knowledge Bases”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2017.

[470] M. Qu, X. Ren, Y. Zhang, and J. Han. “Weakly-supervised Relation Extraction by Pattern-enhanced Embedding Learning”. In: *The Web Conference (WWW)*. 2018.

[471] K. Raghunathan, H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. D. Manning. “A Multi-Pass Sieve for Coreference Resolution”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2010.

[472] E. Rahm and H. H. Do. “Data Cleaning: Problems and Current Approaches”. *IEEE Data Engineering Bulletin*. 23(4): 3–13. 2000.

[473] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. “SQuAD: 100, 000+ Questions for Machine Comprehension of Text”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2016.

[474] H. Rashkin, E. Choi, J. Y. Jang, S. Volkova, and Y. Choi. “Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017. 2931–2937.

[475] V. Rastogi, N. N. Dalvi, and M. N. Garofalakis. “Large-Scale Collective Entity Matching”. *Proceedings of the VLDB Endowment*. 4(4): 208–218. 2011.

[476] L. Ratinov and D. Roth. “Design Challenges and Misconceptions in Named Entity Recognition”. In: *Conference on Computational Natural Language Learning (CoNLL)*. 2009.

[477] L. Ratinov, D. Roth, D. Downey, and M. Anderson. “Local and Global Algorithms for Disambiguation to Wikipedia”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2011.

[478] D. Ravichandran and E. H. Hovy. “Learning surface text patterns for a Question Answering System”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2002.

[479] S. Razniewski and P. Das. “Structured Knowledge: Have We Made Progress? An Extrinsic Study of KB Coverage over 19 Years”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2020.

[480] S. Razniewski, N. Jain, P. Mirza, and G. Weikum. “Coverage of Information Extraction from Sentences and Paragraphs”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019.

[481] S. Razniewski, F. Korn, W. Nutt, and D. Srivastava. “Identifying the Extent of Completeness of Query Answers over Partially Complete Databases”. In: *ACM Conference on Management of Data (SIGMOD)*. 2015.

[482] S. Razniewski, F. M. Suchanek, and W. Nutt. “But What Do We Actually Know?” In: *Workshop on Automated Knowledge Base Construction (AKBC)*. 2016.

[483] T. Rebele, F. M. Suchanek, J. Hoffart, J. A. Biega, E. Kuzey, and G. Weikum. “YAGO: a multilingual knowledge base from Wikipedia, Wordnet, and Geonames ”. In: *International Semantic Web Conference (ISWC)*. 2016.

[484] R. Reinanda, E. Meij, and M. de Rijke. “Knowledge Graphs: An Information Retrieval Perspective”. *Foundations and Trends in Information Retrieval*. 2020.

[485] F. Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, and S. Vaithyanathan. “An Algebraic Approach to Rule-Based Information Extraction”. In: *IEEE International Conference on Data Engineering (ICDE)*. 2008.

[486] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. “HoloClean: Holistic Data Repairs with Probabilistic Inference”. *Proceedings of the VLDB Endowment*. 10(11): 1190–1201. 2017.

[487] X. Ren, J. Shen, M. Qu, X. Wang, Z. Wu, Q. Zhu, M. Jiang, F. Tao, S. Sinha, D. Liem, P. Ping, R. M. Weinshilboum, and J. Han. “Life-iNet: A Structured Network-Based Knowledge Exploration and Analytics System for Life Sciences”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2017.

[488] X. Ren, Z. Wu, W. He, M. Qu, C. R. Voss, H. Ji, T. F. Abdelzaher, and J. Han. “CoType: Joint Extraction of Typed Entities and Relations with Knowledge Bases”. In: *The Web Conference (WWW)*. 2017.

[489] M. Richardson and P. M. Domingos. “Markov logic networks”. *Machine Learning*. 62(1-2): 107–136. 2006.

[490] S. Riedel. “Improving the Accuracy and Efficiency of MAP Inference for Markov Logic”. *CoRR*. abs/1206.3282. 2012.

[491] S. Riedel, L. Yao, and A. McCallum. “Modeling Relations and Their Mentions without Labeled Text”. In: *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. 2010.

[492] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin. “Relation Extraction with Matrix Factorization and Universal Schemas”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2013.

[493] D. Ritze and C. Bizer. “Matching Web Tables To DBpedia - A Feature Utility Study”. In: *Conference on Extending Database Technology (EDBT)*. 2017.

[494] M. Röder, R. Usbeck, and A. N. Ngomo. “GERBIL - Benchmarking Named Entity Recognition and Linking consistently”. *Semantic Web Journal (SWJ)*. 9(5): 605–625. 2018.

[495] J. Romero, S. Razniewski, K. Pal, J. Z. Pan, A. Sakhadeo, and G. Weikum. “Commonsense Properties from Query Logs and Question Answering Forums”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2019. 1411–1420.

[496] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming*. Vol. 2. Elsevier, 2006.

[497] D. Roth. “On the Hardness of Approximate Reasoning”. *Artificial Intelligence (AI)*. 82(1-2): 273–302. 1996.

[498] D. Roth and V. Srikumar. “Integer Linear Programming Formulations in Natural Language Processing (Tutorial Materials)”. In: *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. 2017.

[499] D. Roth and W.-t. Yih. “Global inference for entity and relation identification via a linear programming formulation”. *Introduction to statistical relational learning*: 553–580. 2007.

[500] D. Roth and W. Yih. “Integer linear programming inference for conditional random fields”. In: *International Conference on Machine Learning (ICML)*. 2005.

[501] M. Rotmensch, Y. Halpern, A. Tlimat, S. Horng, and D. Sontag. “Learning a health knowledge graph from electronic medical records”. *Scientific reports*. 7(1): 1–11. 2017.

[502] S. Roy, T. Vieira, and D. Roth. “Reasoning about Quantities in Natural Language”. *Transactions of the Association for Computational Linguistics (TACL)*. 3: 1–13. 2015.

[503] D. Ruffinelli, S. Broscheit, and R. Gemulla. “You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings”. In: *International Conference on Learning Representations (ICLR)*. 2020.

[504] S. J. Russell and P. Norvig. *Artificial intelligence - a modern approach, 2nd Edition*. Prentice Hall, 2003.

[505] A. Sabharwal and H. Sedghi. “How Good Are My Predictions? Efficiently Approximating Precision-Recall Curves for Massive Datasets”. In: *Conference on Uncertainty in Artificial Intelligence (UAI)*. 2017.

[506] S. K. Sahu, F. Christopoulou, M. Miwa, and S. Ananiadou. “Inter-sentence Relation Extraction with Document-level Graph Convolutional Neural Network”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019.

[507] A. Sahuguet and F. Azavant. “Building Light-Weight Wrappers for Legacy Web Data-Sources Using W4F”. In: *Conference on Very Large Databases (VLDB)*. 1999.

[508] H. Samet, J. Sankaranarayanan, M. D. Lieberman, M. D. Adelfio, B. C. Fruin, J. M. Lotkowski, D. Panozzo, J. Sperling, and B. E. Teitler. “Reading news with maps by exploiting spatial synonyms”. *Communication of the ACM*. 57(10): 64–77. 2014.

[509] C. N. dos Santos, B. Xiang, and B. Zhou. "Classifying Relations by Ranking with Convolutional Neural Networks". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2015.

[510] M. Sap, R. L. Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith, and Y. Choi. "ATOMIC: An Atlas of Machine Commonsense for If-Then Reasoning". In: *Conference on Artificial Intelligence (AAAI)*. 2019.

[511] S. Sarawagi *et al.* "Information extraction". *Foundations and Trends in Databases*. 1(3): 261–377. 2008.

[512] S. Sarawagi and S. Chakrabarti. "Open-domain quantity queries on web tables: annotation, response, and consensus models". In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2014. 711–720.

[513] U. Sawant and S. Chakrabarti. "Learning joint query interpretation and response ranking". In: *The Web Conference (WWW)*. 2013.

[514] H. Saxena, L. Golab, and I. F. Ilyas. "Distributed Implementations of Dependency Discovery Algorithms". *Proceedings of the VLDB Endowment*. 12(11): 1624–1636. 2019.

[515] J. Schmidhuber. "Deep learning in neural networks: An overview". *Neural Networks*. 61: 85–117. 2015.

[516] A. Schrijver. *Theory of linear and integer programming. Wiley-Interscience series in discrete mathematics and optimization*. Wiley, 1999.

[517] L. Schubert. "Can we derive general world knowledge from texts". In: *International Conference on Human Language Technology (HLT)*. 2002.

[518] özge Sevgili, A. Panchenko, and C. Biemann. "Improving Neural Entity Disambiguation with Graph Embeddings". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019.

[519] J. Shang, L. Liu, X. Gu, X. Ren, T. Ren, and J. Han. "Learning Named Entity Tagger using Domain-Specific Dictionary". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.

[520] J. Shang, X. Zhang, L. Liu, S. Li, and J. Han. "NetTaxo: Automated Topic Taxonomy Construction from Text-Rich Network". In: *The Web Conference (WWW)*. 2020. 1908–1919.

[521] W. Shen, P. DeRose, R. McCann, A. Doan, and R. Ramakrishnan. "Toward best-effort information extraction". In: *ACM Conference on Management of Data (SIGMOD)*. 2008.

[522] W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. "Declarative Information Extraction Using Datalog with Embedded Extraction Predicates". In: *Conference on Very Large Databases (VLDB)*. 2007.

[523] W. Shen, J. Wang, and J. Han. “Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions”. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 27(2): 443–460. 2015.

[524] W. Shen, J. Wang, P. Luo, and M. Wang. “LIEGE: : link entities in web lists with knowledge base”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2012.

[525] W. Shen, J. Wang, P. Luo, and M. Wang. “LINDEN: linking named entities with knowledge base via semantic knowledge”. In: *The Web Conference (WWW)*. 2012.

[526] W. Shen, J. Wang, P. Luo, and M. Wang. “Linking named entities in Tweets with knowledge base via user interest modeling”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2013.

[527] J. Shin, S. Wu, F. Wang, C. D. Sa, C. Zhang, and C. Ré. “Incremental Knowledge Base Construction Using DeepDive”. *Proceedings of the VLDB Endowment*. 8(11): 1310–1321. 2015.

[528] J. Shinavier, K. Branson, W. Zhang, S. Dastgheib, Y. Gao, B. G. Arsintescu, F. özcan, and E. Meij. “Panel: Knowledge Graph Industry Applications”. In: *The Web Conference (WWW)*. 2019.

[529] V. Shwartz, Y. Goldberg, and I. Dagan. “Improving Hypernymy Detection with an Integrated Path-based and Distributional Method”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2016.

[530] A. Sil and S. Cucerzan. “Towards Temporal Scoping of Relational Facts based on Wikipedia Data”. In: *Conference on Computational Natural Language Learning (CoNLL)*. 2014.

[531] A. Sil, H. Ji, D. Roth, and S. Cucerzan. “Multi-lingual Entity Discovery and Linking”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2018.

[532] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu. “Open mind common sense: Knowledge acquisition from the general public”. In: *OTM Confederated International Conferences*. 2002.

[533] R. Singh, V. V. Meduri, A. K. Elmagarmid, S. Madden, P. Papotti, J. Quiané-Ruiz, A. Solar-Lezama, and N. Tang. “Synthesizing Entity Matching Rules by Examples”. *Proceedings of the VLDB Endowment*. 11(2): 189–202. 2017.

[534] S. Singh, A. Subramanya, F. C. N. Pereira, and A. McCallum. “Large-Scale Cross-Document Coreference Using Distributed Inference and Hierarchical Models”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2011.

[535] S. Singh, A. Subramanya, F. Pereira, and A. McCallum. “Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia”. *University of Massachusetts, Amherst, Technical Report UM-CS-2012*. 15. 2012.

[536] A. Singhal. “Introducing the Knowledge Graph: things, not strings”. In: *Google Blog, 16 May 2012, https://www.blog.google/products/search/introducing-knowledge-graph-things-not/*.

[537] P. Singla and P. M. Domingos. “Entity Resolution with Markov Logic”. In: *IEEE Conference on Data Mining (ICDM)*. 2006.

[538] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B. P. Hsu, and K. Wang. “An Overview of Microsoft Academic Service (MAS) and Applications”. In: *The Web Conference (WWW)*. 2015.

[539] S. Sizov, M. Theobald, S. Siersdorfer, G. Weikum, J. Graupmann, M. Biwer, and P. Zimmer. “The BINGO! System for Information Portal Generation and Expert Web Search”. In: *Conference on Innovative Data Systems Research (CIDR)*. 2003.

[540] P. Smeros, C. Castillo, and K. Aberer. “SciLens: Evaluating the Quality of Scientific News Articles Using Social Media and Scientific Literature Indicators”. In: *The Web Conference (WWW)*. 2019.

[541] R. Snow, D. Jurafsky, and A. Y. Ng. “Semantic Taxonomy Induction from Heterogenous Evidence”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2006.

[542] L. B. Soares, N. FitzGerald, J. Ling, and T. Kwiatkowski. “Matching the Blanks: Distributional Similarity for Relation Learning”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019.

[543] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. “Reasoning With Neural Tensor Networks for Knowledge Base Completion”. In: *Neural Information Processing Systems (NeurIPS)*. 2013.

[544] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. “Semantic Compositionality through Recursive Matrix-Vector Spaces”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2012.

[545] S. Soderland. “Learning Information Extraction Rules for Semi-Structured and Free Text”. *Machine Learning*. 34(1-3): 233–272. 1999.

[546] D. Sorokin and I. Gurevych. “Context-Aware Representations for Knowledge Base Relation Extraction”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017.

[547] A. Soulet, A. Giacometti, B. Markhoff, and F. M. Suchanek. “Representativeness of Knowledge Bases with the Generalized Benford’s Law”. In: *International Semantic Web Conference (ISWC)*. 2018.

[548] R. Speer and C. Havasi. “Representing General Relational Knowledge in ConceptNet 5”. In: *Conference on Language Resources and Evaluation (LREC)*. 2012.

[549] R. Speer, J. Chin, and C. Havasi. “ConceptNet 5.5: An Open Multilingual Graph of General Knowledge”. In: *Conference on Artificial Intelligence (AAAI)*. 2017.

[550] V. I. Spitkovsky and A. X. Chang. “A Cross-Lingual Dictionary for English Wikipedia Concepts”. In: *Conference on Language Resources and Evaluation (LREC)*. 2012.

[551] S. Staab and R. Studer. *Handbook on Ontologies*. Springer, 2009.

[552] C. Stab and I. Gurevych. “Parsing Argumentation Structures in Persuasive Essays”. *Computational Linguistics*. 43(3): 619–659. 2017.

[553] G. Stanovsky, J. Michael, L. Zettlemoyer, and I. Dagan. “Supervised Open Information Extraction”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[554] D. Stepanova, M. H. Gad-Elrab, and V. T. Ho. “Rule Induction and Reasoning over Knowledge Graphs”. In: *Reasoning Web Summer School*. Springer, 2018. 142–172.

[555] J. Strötgen, M. Gertz, G. Hirst, and R. Huang. “Domain-Sensitive Temporal Tagging”. *Computational Linguistics*. 44(2). 2018.

[556] J. Strötgen, T. Tran, A. Friedrich, D. Milchevski, F. Tomazic, A. Marusczyk, H. Adel, D. Stepanova, F. Hildebrand, and E. Kharlamov. “Towards the Bosch Materials Science Knowledge Base”. In: *International Semantic Web Conference (ISWC)*. 2019.

[557] M. Strube and S. P. Ponzetto. “WikiRelate! Computing Semantic Relatedness Using Wikipedia”. In: *National Conference on Artificial Intelligence*. 2006.

[558] F. M. Suchanek. “The Need to Move beyond Triples”. In: *Workshop on Narrative Extraction From Texts (Text2Story), co-located with ECIR Conference*. 2020.

[559] F. M. Suchanek, S. Abiteboul, and P. Senellart. “PARIS: Probabilistic Alignment of Relations, Instances, and Schema”. *Proceedings of the VLDB Endowment*. 5(3): 157–168. 2011.

[560] F. M. Suchanek, G. Ifrim, and G. Weikum. “Combining linguistic and statistical analysis to extract relations from web documents”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2006.

[561] F. M. Suchanek, G. Ifrim, and G. Weikum. “LEILA: Learning to Extract Information by Linguistic Analysis”. In: *Ontology Learning and Population*. 2006.

[562] F. M. Suchanek, G. Kasneci, and G. Weikum. “Yago: a core of semantic knowledge”. In: *The Web Conference (WWW)*. 2007.

[563] F. M. Suchanek, G. Kasneci, and G. Weikum. “ Yago - A Large Ontology from Wikipedia and WordNet ”. In: *Journal of Web Semantics (JWS)*. 2008.

[564] F. M. Suchanek, J. Lajus, A. Boschin, and G. Weikum. “Knowledge Representation and Rule Mining in Entity-Centric Knowledge Bases”. In: *Reasoning Web Summer School*. Springer, 2019.

[565] F. M. Suchanek and N. Preda. “Semantic Culturomics (vision paper)”. *Proceedings of the VLDB Endowment*. 7(12). 2014.

[566] F. M. Suchanek, M. Sozio, and G. Weikum. “SOFIE: a self-organizing framework for information extraction”. In: *The Web Conference (WWW)*. 2009.

[567] T. Sun, C. Zhang, Y. Ji, and Z. Hu. “Reinforcement Learning for Distantly Supervised Relation Extraction”. *IEEE Access*. 7: 98023–98033. 2019.

[568] M. Surdeanu, D. McClosky, J. Tibshirani, J. Bauer, A. X. Chang, V. I. Spitkovsky, and C. D. Manning. “A Simple Distant Supervision Approach for the TAC-KBP Slot Filling Task”. In: *Text Analysis Conference (TAC)*. 2010.

[569] C. Sutton, A. McCallum, *et al.* “An introduction to conditional random fields”. *Foundations and Trends in Machine Learning*. 4(4): 267–373. 2012.

[570] R. Takanobu, T. Zhang, J. Liu, and M. Huang. “A Hierarchical Framework for Relation Extraction with Reinforcement Learning”. In: *Conference on Artificial Intelligence (AAAI)*. 2019.

[571] P. P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. C. N. Pereira. “Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2008.

[572] P. P. Talukdar, D. Wijaya, and T. M. Mitchell. “Acquiring temporal constraints between relations”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2012.

[573] P. P. Talukdar, D. Wijaya, and T. M. Mitchell. “Coupled temporal scoping of relational facts”. In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2012.

[574] C. H. Tan, E. Agichtein, P. Ipeirotis, and E. Gabrilovich. “Trust, but verify: predicting contribution quality for knowledge base construction and curation”. In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2014.

[575] N. Tandon, G. de Melo, A. De, and G. Weikum. “Knowlywood: Mining Activity Knowledge From Hollywood Narratives”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2015. 223–232.

[576] N. Tandon, G. de Melo, F. Suchanek, and G. Weikum. “WebChild : Harvesting and Organizing Commonsense Knowledge from the Web”. *ACM Conference on Web Search and Data Mining (WSDM)*. 2014.

[577] B. Taneva, T. Cheng, K. Chakrabarti, and Y. He. “Mining acronym expansions and their meanings using query click log”. In: *The Web Conference (WWW)*. 2013.

[578] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. “ArnetMiner: extraction and mining of academic social networks”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2008. 990–998.

[579] T. P. Tanon, D. Stepanova, S. Razniewski, P. Mirza, and G. Weikum. “Completeness-aware Rule Learning from Knowledge Graphs”. In: *Joint Conference on Artificial Intelligence (IJCAI)*. 2018.

[580] T. P. Tanon and F. Suchanek. “Querying the Edit History of Wikidata”. In: *European Semantic Web Conference (ESWC)*. 2019.

[581] T. P. Tanon, D. Vrandecic, S. Schaffert, T. Steiner, and L. Pintscher. “From Freebase to Wikidata: The Great Migration”. In: *The Web Conference (WWW)*. 2016.

[582] T. P. Tanon, G. Weikum, and F. M. Suchanek. “YAGO 4: A Reason-able Knowledge Base ”. In: *European Semantic Web Conference (ESWC)*. 2020.

[583] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. “FEVER: a Large-scale Dataset for Fact Extraction and VERification”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[584] D. Tikk, P. Thomas, P. Palaga, J. Hakenberg, and U. Leser. “A Comprehensive Benchmark of Kernel Methods to Extract Protein-Protein Interactions from Literature”. *PLoS Computational Biology*. 6(7). 2010.

[585] B. D. Trisedya, G. Weikum, J. Qi, and R. Zhang. “Neural Relation Extraction for Knowledge Base Enrichment”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019.

[586] B. Trushkowsky, T. Kraska, M. J. Franklin, P. Sarkar, and V. Ramachandran. “Crowdsourcing Enumeration Queries: Estimators and Interfaces”. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 27(7): 1796–1809. 2015.

[587] W. Tunstall-Pedoe. “True Knowledge: Open-Domain Question Answering Using Structured Knowledge and Inference”. *AI Magazine*. 31(3): 80–92. 2010.

[588] J. D. Ullman and J. Widom. *A first course in database systems (2nd edition)*. Prentice Hall, 2002.

[589] C. Unger, A. Freitas, and P. Cimiano. “An Introduction to Question Answering over Linked Data”. In: *Reasoning Web Summer School*. Springer, 2014.

[590] N. UzZaman, H. Llorens, L. Derczynski, J. F. Allen, M. Verhagen, and J. Pustejovsky. “SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations”. In: *Workshop on Semantic Evaluation (SemEval)*. 2013.

[591] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. “Attention is All you Need”. In: *Neural Information Processing Systems (NeurIPS)*. 2017.

[592] V. V. Vazirani. *Approximation algorithms*. Springer, 2001.

[593] P. Venetis, A. Y. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, and C. Wu. “Recovering Semantics of Tables on the Web”. *Proceedings of the VLDB Endowment*. 4(9): 528–538. 2011.

[594] P. Verga, D. Belanger, E. Strubell, B. Roth, and A. McCallum. “Multilingual Relation Extraction using Compositional Universal Schema”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2016.

[595] P. Verga, A. Neelakantan, and A. McCallum. “Generalizing to Unseen Entities and Entity Pairs with Row-less Universal Schema”. In: *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. 2017.

[596] P. Verga, E. Strubell, and A. McCallum. “Simultaneously Self-Attending to All Mentions for Full-Abstract Biological Relation Extraction”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[597] M. Verhagen, I. Mani, R. Sauri, J. Littman, R. Knippen, S. B. Jang, A. Rumshisky, J. Phillips, and J. Pustejovsky. “Automating Temporal Annotation with TARSQI”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2005.

[598] K. Vieira, L. Barbosa, A. S. da Silva, J. Freire, and E. S. de Moura. “Finding seeds to bootstrap focused crawlers”. *World Wide Web*. 19(3): 449–474. 2016.

[599] J. Völker and M. Niepert. “Statistical schema induction”. In: *European Semantic Web Conference (ESWC)*. 2011.

[600] D. Vrandecic and M. Krötzsch. “Wikidata: a free collaborative knowledgebase”. *Communication of the ACM*. 57(10): 78–85. 2014.

[601] W3C. *OWL 2 Web Ontology Language Document Overview (Second Edition), https://www.w3.org/TR/owl2-overview/*. World Wide Web Consortium, 2012.

[602] W3C. *RDF 1.1 Primer, https://www.w3.org/TR/rdf11-primer/*. World Wide Web Consortium, 2014.

[603] W3C. *RDF Primer, https://www.w3.org/TR/2004/REC-rdf-primer-20040210/*. World Wide Web Consortium, 2004.

[604] W3C. *RDF Schema, https://www.w3.org/TR/rdf-schema/*. World Wide Web Consortium, 2014.

[605] A. Waagmeester, G. Stupp, S. Burgstaller-Muehlbacher, B. M. Good, M. Griffith, O. L. Griffith, K. Hanspers, H. Hermjakob, T. S. Hudson, K. Hybiske, *et al.* “Wikidata as a knowledge graph for the life sciences”. *ELife*. 9. 2020.

[606] D. Wadden, K. Lo, L. L. Wang, S. Lin, M. van Zuylen, A. Cohan, and H. Hajishirzi. “Fact or Fiction: Verifying Scientific Claims”. *arXiv:2004.14974*. 2020.

[607] H. Wan, Y. Zhang, J. Zhang, and J. Tang. “AMiner: Search and Mining of Academic Social Networks”. *Data Intelligence*. 1(1): 58–76. 2019.

[608] C. Wang, K. Chakrabarti, Y. He, K. Ganjam, Z. Chen, and P. A. Bernstein. “Concept Expansion Using Web Tables”. In: *The Web Conference (WWW)*. 2015. 1198–1208.

[609] C. Wang, M. Danilevsky, J. Liu, N. Desai, H. Ji, and J. Han. “Constructing Topical Hierarchies in Heterogeneous Information Networks”. In: *IEEE Conference on Data Mining (ICDM)*. 2013. 767–776.

[610] H. Wang, M. Tan, M. Yu, S. Chang, D. Wang, K. Xu, X. Guo, and S. Potdar. “Extracting Multiple-Relations in One-Pass with Pre-Trained Transformers”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019.

[611] Q. Wang, Z. Mao, B. Wang, and L. Guo. "Knowledge Graph Embedding: A Survey of Approaches and Applications". *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 29(12): 2724–2743. 2017.

[612] R. C. Wang and W. W. Cohen. "Iterative Set Expansion of Named Entities Using the Web". In: *IEEE Conference on Data Mining (ICDM)*. 2008. 1091–1096.

[613] X. Wang, X. L. Dong, Y. Li, and A. Meliou. "MIDAS: Finding the right web sources to fill knowledge gaps". In: *ACM Conference on Management of Data (SIGMOD)*. 2019.

[614] Y. Wang, M. Dylla, M. Spaniol, and G. Weikum. "Coupling Label Propagation and Constraints for Temporal Fact Extraction". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2012.

[615] Y. Wang, B. Yang, L. Qu, M. Spaniol, and G. Weikum. "Harvesting facts from textual web sources by constrained label propagation". In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2011.

[616] Z. Wang, J. Zhang, J. Feng, and Z. Chen. "Knowledge Graph and Text Jointly Embedding". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[617] Z. Wang, J. Li, Z. Wang, S. Li, M. Li, D. Zhang, Y. Shi, Y. Liu, P. Zhang, and J. Tang. "XLore: A Large-scale English-Chinese Bilingual Knowledge Graph". In: *International Semantic Web Conference (ISWC)*. 2013.

[618] J. Weeds, D. J. Weir, and D. McCarthy. "Characterising Measures of Lexical Distributional Similarity". In: *Conference on Computational Linguistics (COLING)*. 2004.

[619] G. Weikum. "Entities with Quantities". *IEEE Data Engineering Bulletin*. 43(1): 4–8. 2020.

[620] G. Weikum, J. Hoffart, and F. M. Suchanek. "Knowledge Harvesting: Achievements and Challenges". In: *Computing and Software Science - State of the Art and Perspectives*. Vol. 10000. *Lecture Notes in Computer Science*. Springer, 2019. 217–235.

[621] G. Weikum, J. Hoffart, and F. M. Suchanek. "Ten Years of Knowledge Harvesting: Lessons and Challenges". *IEEE Data Engineering Bulletin*. 39(3): 41–50. 2016.

[622] G. Weikum and M. Theobald. "From information to knowledge: harvesting entities and relationships from web sources". In: *ACM Symposium on Principles of Database Systems (PODS)*. 2010.

[623] S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. "Entity resolution with iterative blocking". In: *ACM Conference on Management of Data (SIGMOD)*. 2009.

[624] M. L. Wick, K. Rohanimanesh, K. Schultz, and A. McCallum. “A unified approach for schema matching, coreference and canonicalization”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2008.

[625] F. Wu and D. S. Weld. “Automatically refining the wikipedia infobox ontology”. In: *The Web Conference (WWW)*. 2008.

[626] F. Wu and D. S. Weld. “Autonomously semantifying wikipedia”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2007.

[627] F. Wu and D. S. Weld. “Open Information Extraction Using Wikipedia”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2010.

[628] J. Wu, K. M. Williams, H. Chen, M. Khabsa, C. Caragea, S. Tuarob, A. Ororbia, D. Jordan, P. Mitra, and C. L. Giles. “CiteSeerX: AI in a Digital Library Search Engine”. *AI Magazine*. 36(3): 35–48. 2015.

[629] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer. “Zero-shot Entity Linking with Dense Entity Retrieval”. *CoRR*. abs/1911.03814. 2019.

[630] S. Wu and Y. He. “Enriching Pre-trained Language Model with Entity Information for Relation Classification”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2019.

[631] W. Wu, H. Li, H. Wang, and K. Q. Zhu. “Probase: a probabilistic taxonomy for text understanding”. In: *ACM Conference on Management of Data (SIGMOD)*. 2012.

[632] Y. Wu, D. Bamman, and S. J. Russell. “Adversarial Training for Relation Extraction”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017.

[633] D. Xu, C. Ruan, E. Körpeoglu, S. Kumar, and K. Achan. “Product Knowledge Graph Embedding for E-commerce”. In: *ACM Conference on Web Search and Data Mining (WSDM)*. 2020.

[634] H. Xu, W. Wang, X. Mao, X. Jiang, and M. Lan. “Scaling up Open Tagging from Tens to Thousands: Comprehension Empowered Attribute Value Extraction from Product Title”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019.

[635] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin. “Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2015.

[636] M. Yahya, S. Whang, R. Gupta, and A. Halevy. “Renoun: Fact extraction for nominal attributes”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[637] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. “InfoGather: entity augmentation and attribute discovery by holistic matching with web tables”. In: *ACM Conference on Management of Data (SIGMOD)*. 2012.

[638] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Takefuji. "Wikipedia2Vec: An Optimized Tool for Learning Embeddings of Words and Entities from Wikipedia". *CoRR*. abs/1812.06280. 2018.

[639] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji. "Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation". In: *Conference on Computational Natural Language Learning (CoNLL)*. 2016.

[640] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji. "Learning Distributed Representations of Texts and Entities from Knowledge Base". *Transactions of the Association for Computational Linguistics (TACL)*. 5: 397–411. 2017.

[641] Z. Yang and E. Nyberg. "Leveraging Procedural Knowledge for Task-oriented Search". In: *ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 2015.

[642] L. Yao, A. Haghighi, S. Riedel, and A. McCallum. "Structured relation discovery using generative models". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2011.

[643] L. Yao, S. Riedel, and A. McCallum. "Collective Cross-Document Relation Extraction Without Labelled Data". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2010.

[644] L. Yao, S. Riedel, and A. McCallum. "Universal schema for entity type prediction". In: *Workshop on Automated Knowledge Base Construction (AKBC)*. 2013.

[645] Y. Yao, D. Ye, P. Li, X. Han, Y. Lin, Z. Liu, Z. Liu, L. Huang, J. Zhou, and M. Sun. "DocRED: A Large-Scale Document-Level Relation Extraction Dataset". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019.

[646] A. Yates and O. Etzioni. "Unsupervised Methods for Determining Object and Relation Synonyms on the Web". *Journal of Artificial Intelligence Research*. 34: 255–296. 2009.

[647] J. Yeo, H. Cho, J. Park, and S. Hwang. "Multimodal KB Harvesting for Emerging Spatial Entities". *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 29(5): 1073–1086. 2017.

[648] X. Yin, J. Han, and P. S. Yu. "Truth Discovery with Multiple Conflicting Information Providers on the Web". *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 20(6): 796–808. 2008.

[649] R. Yu, U. Gadiraju, B. Fetahu, O. Lehmberg, D. Ritze, and S. Dietze. "KnowMore - knowledge base augmentation with structured web markup". *Semantic Web Journal (SWJ)*. 10(1): 159–180. 2019.

[650] Y. Yuan, L. Liu, S. Tang, Z. Zhang, Y. Zhuang, S. Pu, F. Wu, and X. Ren. "Cross-Relation Cross-Bag Attention for Distantly-Supervised Relation Extraction". In: *Conference on Artificial Intelligence (AAAI)*. 2019.

[651] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi. “From Recognition to Cognition: Visual Commonsense Reasoning”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[652] D. Zeng, K. Liu, Y. Chen, and J. Zhao. “Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2015.

[653] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao. “Relation Classification via Convolutional Deep Neural Network”. In: *Conference on Computational Linguistics (COLING)*. 2014.

[654] Q. Zeng, J. M. Patel, and D. Page. “QuickFOIL: Scalable Inductive Logic Programming”. *Proceedings of the VLDB Endowment*. 8(3): 197–208. 2014.

[655] C. Zhang, C. Ré, M. J. Cafarella, J. Shin, F. Wang, and S. Wu. “DeepDive: declarative knowledge base construction”. *Communication of the ACM*. 60(5): 93–102. 2017.

[656] C. Zhang, J. Shin, C. Ré, M. J. Cafarella, and F. Niu. “Extracting Databases from Dark Data with DeepDive”. In: *ACM Conference on Management of Data (SIGMOD)*. 2016.

[657] D. Zhang, S. Mukherjee, C. Lockard, L. Dong, and A. McCallum. “OpenKI: Integrating Open Information Extraction and Knowledge Bases with Relation Inference”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2019.

[658] M. Zhang and K. Chakrabarti. “InfoGather+: semantic matching and annotation of numeric and time-varying attributes in web tables”. In: *ACM Conference on Management of Data (SIGMOD)*. 2013.

[659] S. Zhang, R. Rudinger, K. Duh, and B. V. Durme. “Ordinal Common-sense Inference”. *Transactions of the Association for Computational Linguistics (TACL)*. 5: 379–395. 2017.

[660] S. Zhang and K. Balog. “Web Table Extraction, Retrieval, and Augmentation: A Survey”. *ACM Transactions on Intelligent Systems Technology (TIST)*. 11(2): 13:1–13:35. 2020.

[661] S. Zhang, E. Meij, K. Balog, and R. Reinanda. “Novel Entity Discovery from Web Tables”. In: *The Web Conference (WWW)*. 2020.

[662] Y. Zhang, Z. G. Ives, and D. Roth. “Evidence-based Trustworthiness”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019.

[663] Y. Zhang, Q. Chen, Z. Yang, H. Lin, and Z. Lu. “BioWordVec, improving biomedical word embeddings with subword information and MeSH”. *Scientific Data*. 6: 52. 2019.

[664] Y. Zhang and X. Chen. “Explainable Recommendation: A Survey and New Perspectives”. *Foundations and Trends in Information Retrieval*. 14(1): 1–101. 2020.

[665] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning. “Position-aware Attention and Supervised Data Improve Slot Filling”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017.

[666] G. Zheng, S. Mukherjee, X. L. Dong, and F. Li. “OpenTag: Open Attribute Value Extraction from Product Profiles”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2018.

[667] S. Zheng, R. Song, J. Wen, and C. L. Giles. “Efficient record-level wrapper induction”. In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2009.

[668] S. Zheng, R. Song, J. Wen, and D. Wu. “Joint optimization of wrapper generation and template detection”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2007.

[669] Z. Zheng, F. Li, M. Huang, and X. Zhu. “Learning to Link Entities with Knowledge Base”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2010.

[670] G. Zhou, M. Zhang, D. Ji, and Q. Zhu. “Tree Kernel-Based Relation Extraction with Context-Sensitive Structured Parse Tree Information”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2007.

[671] E. Zhu, D. Deng, F. Nargesian, and R. J. Miller. “JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes”. In: *ACM Conference on Management of Data (SIGMOD)*. 2019.

[672] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J. Wen. “StatSnowball: a statistical approach to extracting entity relationships”. In: *The Web Conference (WWW)*. 2009.

[673] J. Zhu, Z. Nie, J. Wen, B. Zhang, and W. Ma. “Simultaneous record detection and attribute labeling in web data extraction”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2006.

[674] J. Zhu, B. Zhang, Z. Nie, J. Wen, and H. Hon. “Webpage understanding: an integrated approach”. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2007.

[675] Q. Zhu, H. Wei, B. Sisman, D. Zheng, C. Faloutsos, X. L. Dong, and J. Han. “Collective Multi-type Entity Alignment Between Knowledge Graphs”. In: *The Web Conference (WWW)*. 2020. 2241–2252.

[676] X. Zhu and Z. Ghahramani. “Learning from labeled and unlabeled data with label propagation”. *Technical Report, Carnegie Mellon University*. 2002.

[677] S. Zwicklbauer, C. Seifert, and M. Granitzer. “Robust and Collective Entity Disambiguation through Semantic Embeddings”. In: *ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 2016.