

From Information to Knowledge: Harvesting Entities and Relationships from Web Sources

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

Martin Theobald
Max Planck Institute for Informatics
Saarbrücken, Germany
mtb@mpi-inf.mpg.de

ABSTRACT

There are major trends to advance the functionality of search engines to a more expressive semantic level. This is enabled by the advent of knowledge-sharing communities such as Wikipedia and the progress in automatically extracting entities and relationships from semistructured as well as natural-language Web sources. Recent endeavors of this kind include DBpedia, EntityCube, KnowItAll, ReadTheWeb, and our own YAGO-NAGA project (and others). The goal is to automatically construct and maintain a comprehensive knowledge base of facts about named entities, their semantic classes, and their mutual relations as well as temporal contexts, with high precision and high recall. This tutorial discusses state-of-the-art methods, research opportunities, and open challenges along this avenue of knowledge harvesting.

Categories and Subject Descriptors

H.1.0 [Information Systems]: Models and Principles—General

General Terms

Design, Management

Keywords

Knowledge Harvesting, Information Extraction, Entities, Relationships

1. INTRODUCTION

1.1 Motivation and Potential Benefits

The Web bears the potential of being the world's greatest encyclopedic source, but we are far from exploiting this potential. Valuable scientific and cultural content is all mixed up with huge amounts of noisy, low-quality, unstructured text and media. However, the proliferation of knowledge-sharing communities like Wikipedia and the advances in automated information extraction from Web pages open up an unprecedented opportunity: can we systematically harvest facts from the Web and compile them into a compre-

hensive machine-readable knowledge base about the world's entities, their semantic properties, and their relationships with each other. Imagine a "Structured Wikipedia" that has the same scale and richness as Wikipedia itself but offers a precise and concise representation of knowledge, e.g., in the RDF format. This would enable expressive and highly precise querying, e.g., in the SPARQL language (or appropriate extensions), with additional capabilities for informative ranking of query results.

The benefits from solving the above challenge would be enormous. Potential applications include 1) a formalized *machine-readable encyclopedia* that can be queried with high precision like a semantic database; 2) a key asset for *disambiguating entities* by supporting fast and accurate mappings of textual phrases onto named entities in the knowledge base; 3) an enabler for entity-relationship-oriented *semantic search* on the Web, for detecting entities and relations in Web pages and reasoning about them in expressive (probabilistic) logics; 4) a backbone for *natural-language question answering* that would aid in dealing with entities and their relationships in answering who/where/when/ etc. questions; 5) a key asset for *machine translation* (e.g., English to German) and interpretation of spoken dialogs, where world knowledge provides essential context for disambiguation; 6) a *catalyst for acquisition of further knowledge* and largely automated maintenance and growth of the knowledge base.

While these application areas cover a broad, partly AI-flavored ground, the most notable one from a database perspective is semantic search: finally bringing DB methodology to Web search! For example, users (or tools on behalf of users) would be able to formulate queries about succulents that grow both in Africa and America, politicians who are also scientists or are married to singers, or flu medication that can be taken by people with high blood pressure. And the search engine would return precise and concise answers: lists of entities or entity pairs (depending on the question structure), for example, Angela Merkel, Benjamin Franklin, etc., or Nicolas Sarkozy for the questions about scientists. This would be a quantum leap over today's search where answers are embedded if not buried in lots of result pages, and the human would have to read them to extract entities and connect them to other entities. In this sense, the envisioned large-scale *knowledge harvesting* [137] from Web sources may also be viewed as *machine reading* [29, 58, 60].

Of course, this vision is not new. Universal knowledge bases have been an AI objective since the pioneering work on Cyc [87, 88] and the early enthusiasm about the original roadmap for the Semantic Web [19, 120]. But there are much more favorable assets to start with available today, and a fair number of ongoing endeavors, both in academia and industrial research, have embarked on the mission of building large knowledge bases. *freebase.com* and *trueknowledge.com* are compiling huge amounts of entity-relationship-oriented

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PODS'10, June 6–11, 2010, Indianapolis, Indiana, USA.
Copyright 2010 ACM 978-1-4503-0033-9/10/06 ...\$10.00.

facts, the community endeavor DBpedia [12] (*dbpedia.org*) which is harvesting RDF subject-property-object triples from Wikipedia and similar sources, the KnowItAll [59] project and its local siblings TextRunner [142], Kylin/KOG [138, 139, 140] and Omnivore [24], which aim to extract arbitrary relations from natural-language texts, the ReadTheWeb [29] team with the ambitious goal of “macro-reading” arbitrary Web pages, the *sig.ma* engine [130] which taps on “triplified” RDF data on the Web, as well as our own project YAGO project [123, 124] which integrates relational knowledge from Wikipedia with the WordNet taxonomy. Further services along these lines are being pursued for specific communities, such as DBLife [49] (*dblife.cs.wisc.edu*) for database research (based on the general-purpose methodology of Cimple [51]) or MedIE [94] for the biomedical domain [57]. In addition, there is an emerging new brand of semantic-search and knowledge-discovery engines that are centered around large knowledge repositories. Representatives include *wolframalpha.com* which computes knowledge answers from a set of hand-crafted databases, *google.com/squared* which arranges search results in a tabular form with entities and attributes, *entitycube.research.microsoft.com* which provides dynamically gathered facts about named entities (based on the StatSnowball methodology [144]), *opencalais.com* which provides services to superimpose structure on documents or Web pages, or *kosmix.com* which uses a large ontology for categorizing questions and identifying entities that are related to the user’s input.

1.2 Scope and Outline of the Tutorial

This tutorial gives an overview of this research avenue, and will go into some depth on specific issues, aiming to identify interesting research problems (for students) and major challenges (for larger teams or institutes).

A knowledge base of the envisioned kind would go through various stages of a life-cycle:

- 1) *building* a large collection of facts about entities, classes, and relationships based on Web sources (or enterprise information, digital library documents, etc.);
- 2) *cleaning* the knowledge base by assessing the validity of facts, possibly removing invalid pieces or completing missing pieces by deduction;
- 3) *querying* the fact collection to answer advanced questions by knowledge workers;
- 4) *ranking* answers to complex queries or reasoning tasks, by means of statistics about informativeness, confidence, compactness, and other criteria;
- 5) *reasoning* about entities and relationships to derive additional knowledge that is not readily given in extensional form;
- 6) *maintaining and growing* the knowledge base as the world evolves, with new facts coming into existence and being digitally expressed on the Internet.

This tutorial will focus on the stages 1, 2, and 6: building, cleaning, maintaining and growing. The second stage, cleaning, is indeed closely intertwined with the other two. Here, reasoning also plays an important role. As for querying, we merely point to languages like SPARQL, which is roughly a schema-free equivalent of select-project-join queries in the relational algebra. There is considerable research on extending SPARQL by reachability constructs, regular expressions, or temporal predicates (e.g., [6, 8, 72, 107, 127, 136]), and, of course, there are big challenges regarding the optimization and scalability of query processing (e.g., [98]). Ranking the results of queries that yield more answers than a human would want to see has been intensively studied for entity-centric search (e.g.,

finding soccer players who played for FC Barcelona) [31, 38, 62, 75, 81, 99, 102, 116, 132]. For general queries that aim to explore the connections between multiple entities (e.g., co-infection with HIV and tuberculosis in South Africa and treatments with particular drugs) or ask for composite facts (e.g., songs and movies such that the song is played in the movie), recent progress on statistical ranking models is presented in [55]. Finally, reasoning over uncertain data is a huge topic (see, e.g., [16, 47]); we consider only specific approaches that are particularly relevant for knowledge harvesting.

The key technology for tackling the goals of knowledge harvesting is known as *information extraction (IE)*. It provides models, algorithms, and tools for lifting Web pages, text sources, semistructured data such as HTML tables or Wikipedia infoboxes into explicit facts – instances of unary, binary, or higher-arity relations. The prevalent methods are (combinations of) rule-based pattern matching, natural language processing (NLP), and statistical machine learning (ML). The IE field has made enormous progress in recent years and became much more scalable, and also less dependent on human supervision (see [1, 46, 50, 113] and references given there).

Fact extraction can be pursued in an *output-oriented targeted* (“closed”) manner or in an input-oriented generic (“open”) manner. In the case of output-oriented targeted IE, we are driven by a given set of relations for which we would like to gather instances. We are flexible in choosing our sources (e.g., can go only for easier or cleaner sources with high return) and we can exploit redundancy on the Web. Moreover, we have great flexibility regarding how deep natural-language text is analyzed in a demand-driven manner. A typical use case is to find the Alma Mater of as many scientists as possible, using a small set of seed facts for training. In the case of input-oriented generic IE, we are focusing on a given input source (e.g., a particular Web page, news site, or discussion forum) and consider all conceivable relations at once. This approach inevitably requires deep analysis of natural-language text, and it can be successful only if sufficient training data is provided. A typical use case is to automatically annotate news articles and extract as many relations as possible from each news item. In this tutorial, we largely focus on output-oriented targeted IE, as a cornerstone of knowledge harvesting.

Several related areas are out of scope of this tutorial. As the outcome of knowledge harvesting may be non-schematic facts with different degrees of confidence in their validity, there is an obvious connection to the broad areas of uncertain data management and data spaces [65, 71]. However, organizing knowledge bases is a special setting with characteristics that are very different from, for example, sensor data. Thus, we will merely borrow selected methods from the general theme of uncertain databases. Data integration and data exchange [20, 54, 61] are latently relevant, but these lines of research are primarily schema-oriented. In knowledge harvesting, on the other hand, we are more concerned with individual entities and their types and relationships. There is no prescriptive schema, typing is the outcome of the harvesting and cleaning process. Data cleaning [97] (e.g., for data warehouses) is an issue insofar as entity matching and record linkage [42, 83] are a key element for a vital knowledge base. We will look at this specific issue, but disregard data cleaning at the level of entire databases. Finally, text and Web mining provides algorithmic building blocks for our agenda. However, we go beyond the informal discovery tasks such as computing interesting tags or tag clouds from blogs and online communities. Our goal is a rigorously formalized, machine-processable knowledge base.

The tutorial is organized by progressively advancing the style of knowledge that we aim to harvest. We start in Section 3 with gathering individual entities and carefully organizing them into seman-

tic classes (unary predicates). Then we raise the bar in Section 4 by looking at typed relationships between entities, limiting ourselves to instances of binary relations (binary predicates). Finally, we address the temporal validity of facts in Section 5, this way extending our scope to non-binary relations (higher-arity predicates or even higher-order predicates). The discussion of these three levels of knowledge harvesting is preceded by a short primer on technical basics in Section 2.

2. TECHNICAL BACKGROUND

Knowledge harvesting and machine reading heavily rely on statistical machine learning (ML) and, to the extent that arbitrary text inputs are considered, also on natural language processing (NLP). In the following we provide some basic reference points.

Statistical Machine Learning (ML). One of the most fundamental ML models are *classifiers*. These are based on supervised learning, with explicit training data. For example, we could train a classifier to test whether a given substring of a Web table or sentence is a temporal expression or not (including, perhaps, adverbial phrases such as “last Monday”), or whether the whole sentence talks about the CEO of a company. Methods for automatic classification of text inputs are very mature; the main families are probabilistic methods (e.g., Bayesian models, logistic regression, etc.) and discriminative models (e.g., support vector machines). Training a classifier requires manual labeling, which is often the bottleneck; insufficient training data will lead to poor accuracy of the classifier’s predictions. Therefore, some *semi-supervised learning* methods aim to combine sparse training data with large collections of additional, unlabeled data (e.g., unlabeled Web pages).

Another fundamental building block from ML are models for *joint segmentation and labeling*. These operate over sequences of tokens and aim to find meaningful subsequences that will then be labeled. For example, consider bibliographic entries with one or more authors, a title, a source like a conference or journal, publisher, publication date and place, and so on. Such entries exhibit very high diversity, including different orders of their elements, all kinds of abbreviations, book chapters so that authors appear for both the article and the entire book, and so on. We can model the structure and possible orderings of components in a probabilistic automaton and then use statistical learning to infer probabilities that govern the production of realistic bibliographic entries. A learned model along these lines would then process a new, previously unseen, entry by identifying segments of contiguous tokens and labeling each segment by its role: author, title, journal, volume, issue, year, etc. Mature ML methods for this purpose are Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) [113, 126]. Their training procedures merely require a set of typical inputs (bibliographic entries in our example), without any explicit manual labeling. The model parameters are learned by likelihood-optimization techniques, often via dynamic programming or gradient-descent methods. For a new input, the trained model then computes the most likely label sequence (or state sequence in the underlying FSA) that explains the input. For HMMs, which use only locally bounded contexts for predicting a label, this is the well-known Viterbi algorithm. For CRFs, which support richer forms of context-aware feature functions, similar but more advanced optimization methods are used.

Natural Language Processing (NLP). Statistical learning is relevant for semistructured inputs such as Web tables or Wikipedia infoboxes as well as unstructured text inputs. In the latter case, there is usually a need to combine ML with NLP techniques [80, 91] (many of which in turn employ ML models and algorithms). The most widely used NLP method is *part-of-speech tagging* (PoS tag-

ging for short). A PoS tag adorns a word in a natural-language sentence with the word’s grammatical role: noun, verb, adjective, preposition, pronoun, etc. This is a shallow technique that relies on a stochastic notion of finite-state automata, often on Hidden Markov Models (HMMs). Training these models is based on likelihood-optimization methods; it requires a large amount of well-formed input sentences but does not need any manual PoS labeling in the training input. Once the model is trained, it can label the words in a new sentence by their PoS tags, using dynamic programming. PoS tagging is not computationally inexpensive, but its run-time cost is usually acceptable and not a showstopper regarding scalability. An important class of tags for the purpose of IE are so-called proper nouns; these are (composite) nouns that do not have a preceding article and are thus candidates for named entities.

For fully understanding the structure of a natural-language sentence, deep parsing is needed. A popular class of parsers are *lexical dependency parsers*, aka. *link parsers*. These are based on probabilistic context-free grammars and compute an edge-labeled planar graph that connects the constituents in the sentence according to the syntactic-logical structure of the sentence. For example, if the second object of a verb (e.g., “gave . . . a present”) is placed far apart from the verb because of a relative clause that elaborates on the first object (e.g., “gave Mary, who had her 18th birthday, a present”), the dependency structure determined by such a deep parser would correctly connect the verb with the first object (“Mary”) and the second object (“a present”). Thus, although dependency parsing is a purely syntax-driven method, it can be seen as a crude approximation of the sentence’s semantics by means of a predicate-argument structure. Unfortunately, dependency parsing is computationally expensive; it typically has to solve dynamic-programming problems over a grammar with a huge number of (probabilistic) production rules. Today, very expressive dependency parsers are available as open-source software (e.g., CMU Link Parser, Stanford Lex Parser, Open NLP tools, etc.).

An active research issue in NLP is to extend this line of analysis into *semantic role labeling* (SRL) [68, 95]. Here a sentence is viewed as a logical predicate, derived from the verbal phrase and perhaps using a trained classifier, that comes with a number of arguments such as acting agent, location, time, purpose, etc. SRL aims to automatically fill these arguments or roles with the proper parts of the sentence. One can think of this as mapping the sentence into a frame-like knowledge representation. For example, sentences that refer to purchases could be mapped into purchase-specific frames with roles like seller, buyer, goods, amount, etc., regardless of whether the sentence is active or passive and whether the verb is “buy”, “acquire”, etc. Obviously this is a very difficult task, especially because there are still only small-scale training corpora available (e.g., PropBank [82] and FrameNet [96]).

Gazetteers. An additional ingredient for PoS tagging and extended forms of semantic role labeling are large dictionaries of person names, location names, temporal expressions, etc. These are often referred to as *gazetteers*, especially in the context of geographic entity names. A popular example are the *GeoNames* gazetteers (see geonames.org; [128] uses it for IE).

3. ENTITIES AND CLASSES

3.1 State of the Art

At this first level of knowledge harvesting, we are interested in collecting all individual entities – persons, companies, cities, products, etc. – and organizing them into semantic classes (types) such as artists, scientists, molecular biologists, singers, guitar players, movies, computer games, etc. Of course, a given entity can belong

to multiple classes; for example, Angela Merkel is in classes like “politicians”, “scientists”, “female persons”, “German chancellors”, and so on. Classes are mutually related by subclass/superclass connections – inclusion dependencies in database jargon and hyponymy/ hypernymy relations in linguistic jargon. For example, “German chancellors” is a subclass of “politicians”. Altogether, we are seeking for no more or less than a comprehensive set of all meaningful entity types, and their instances in the real world: the *isA* and *instanceOf* relations in a universal entity-relationship database. Automatically constructing such a first-level knowledge base is known as *taxonomy induction* or sometimes *ontology learning* (a slight misnomer as ontologies should be richer than mere taxonomies).

Wikipedia as key asset. Ten years ago, this task would have been deemed elusive, being in the realm of “AI-complete” challenges. But today, there are rich assets available that can be harnessed for automatic knowledge harvesting. Most notably, the *WordNet* thesaurus [63] is a hand-crafted collection of more than 100,000 semantic classes along with fairly comprehensive subclass/superclass relations. *WordNet* covers entity types in the form of high-level concepts. It would know that biologists are scientists, scientists are humans, humans are vertebrates, and so on. But it would not know the specialized types of the modern world, such as “laser physicists” or “European commissioners” and it can also not keep up with the emergence of new types in the business and entertainment world, such as “cloud computing providers”, “hedge funds”, “hybrid cars”, “golden raspberry winners”, or “indie rock music”. Moreover, *WordNet* has very little knowledge about instances of its classes. For example, it knows Alan Turing as a scientist, but none of Ted Codd, Jim Gray, Ronald Rivest, or Barbara Liskov. Here is the point where Wikipedia opened up great opportunities. The English version of Wikipedia contains more than 3 million articles, most of which correspond one-to-one to individual entities. In addition, the Wikipedia authors have manually placed the articles in a rich set of categories which can be seen as classes. It may even seem that Wikipedia itself and its category system already constitute a solution to our task. However, this is a treacherous hypothesis. In fact, the category system is very noisy, contains dangling ends and mistakes, and is oriented by association rather than clean taxonomic relations. For example, Robert Sherman (a composer) is in the category “Walt Disney theatricals”, Frank Zappa in “1960s music groups”, and Javier Navarrete in “European composer stubs”, but they are not theatricals, groups, or stubs, respectively. Google is even in the category “Internet history”, but even Google’s harshest critics would not claim that it is history. Similarly, sub-categories do often not reflect the *isA* relation: the category “Einstein family” (in which Albert Einstein is an instance) is a sub-category of “Albert Einstein”, and “books about revolutions” are a sub-category of “revolutions” (which may hold for some books, but certainly not for all of them).

With these considerations, it is not obvious if and to what extent the raw assets of *WordNet* and Wikipedia categories can be harvested and systematically organized. A major breakthrough along these lines has been achieved by the work on building YAGO (Yet Another Great Ontology) [123, 124], the parallel and independent work on WikiTaxonomy [103, 104, 105], and the follow-up work on KOG (Kylin Ontology Generator) [138, 139, 140]. We briefly outline the methodology of YAGO; there are many commonalities with the other two projects. YAGO initializes its class system by importing all *WordNet* classes and their hyponymy/hypernymy (subclass/superclass) relations. Each individual entity that YAGO discovers in Wikipedia needs to be mapped into at least one of the existing YAGO classes. If this fails, the entity (and its related facts)

are not admitted to the knowledge base. To this end, the Wikipedia categories of the entity are compared to names of *WordNet* classes, including the synonyms that *WordNet* knows for a class name. In this comparison and to eliminate non-taxonomic categories, a noun group parser is used to determine the logical head word of the category name. For example, the head word of “American folk music of the 20th century” is “music”. If the head word is in plural or can be cast into plural form (which would not work for “music”), it is attempted to map the head word or the head word with its pre-modifier (e.g., “folk music”) onto a *WordNet* class. If this results in a perfect match of the names, then the entity is added to the *WordNet* class and the category that led to the match is made a subclass of the *WordNet* class. This way, we can automatically make “Presidents of France” a subclass of “Presidents”. There are also limitations of this heuristics: we cannot directly make the connection between the Wikipedia category “Presidents of France” and the *WordNet* class “French” (i.e., French people). The paths in the Wikipedia category system may help to establish this *subclassOf* relation, but then we face many noisy connections as well. Thus, YAGO exploits this potential only to a limited extent.

Overall, these procedures ensure that we can maintain a consistent knowledge base, where consistency eliminates dangling entities or classes and also guarantees that the *subclassOf* relation is acyclic. WikiTaxonomy uses fairly similar heuristics. It additionally considers all instances of a category and a *WordNet* class in deciding whether it should create a *subclassOf* relations. All these methods are tuned towards high precision rather than recall. They are amazingly powerful and work very well, but there are many possible mappings for the taxonomic relations that are not captured this way. The publicly released version of YAGO contains 1,941,578 individual entities in 249,015 classes (plus about 19 million facts – instances of 93 different binary relations carefully extracted from Wikipedia infoboxes). The empirically assessed accuracy of the *instanceOf* and *subclassOf* relations is around 97 percent. YAGO has been integrated into DBpedia and other knowledge bases, and it has been used to create a semantically annotated XML corpus of Wikipedia used as a reference collection in the INEX benchmarking series (see inex.otago.ac.nz).

The more recent work on KOG [140] takes a more ambitious route by considering infobox templates that are abundantly used in Wikipedia. A template is like a noisy form of schema, and the inclusion of one template’s fields (e.g., person with fields *birth_date*, *birth_place*, etc.) in another template (e.g., scientist with additional fields like *alma_mater*, *doctoral_advisor*, *known_for*, etc.) may be viewed as an indicator of a *superClassOf* relation. However, the plethora of infobox templates and their frequent ad-hoc use (with fields like “*filler3*”) makes this kind of knowledge mining a daunting task. KOG uses a cleverly composed suite of machine-learning tools and considers the actual instances of templates to infer subsumption relations. [140] reports encouraging accuracy results, but so far no taxonomic collection has been made available.

Of course, the Web itself is also a huge source of gathering individual entities for given classes. Directly using services such as labs.google.com/sets, by starting with a few seed entities for a class of interest, could have high recall based on surface co-occurrences but would face tremendous noise and thus fail to achieve high precision. Recently, a smarter method has been developed by [134, 135] which exploits semistructured lists at Web scale and carefully ranking the aggregated results by graph-based authority measures. This technique is language-independent (works for Japanese the same way as for English) and achieves high precision, but it remains to be seen whether its recall can go beyond the methods discussed above which exploit hand-crafted sources such as Wikipedia, mo-

vie portals, etc. Moreover, this Web-centric approach needs to start with a given class, it does not determine the classes themselves and their *subclassOf* relations.

Entity disambiguation. When mapping a category to a WordNet class based on a clever name-matching method, a thorny issues that we have so far disregarded is the ambiguity of names. For example, should we map the category “drivers” to the WordNet sense “driver: the operator of a motor vehicle” or to the sense “driver: device driver for a computer” of this polysemous word? YAGO uses a simple heuristics to this end: it always choose the sense that is most frequent in common language according to the WordNet statistics. The same ambiguity issue arises for individual entities as well: do “Dr. Joe Hellerstein, UC Berkeley” and “Prof. Joseph M. Hellerstein, University of California” denote the same person? Conversely, how can we tell that “Michael Jordan”, the NBA player, and “Michael Jordan”, the Berkeley professor, are two different people? Or that “LDA” is an ambiguous acronym, possible meanings being “Latent Dirichlet Allocation”, “Legal Drinking Age”, and others? This raises the general issue of *entity resolution*, also known as entity reconciliation or record linkage. Given a string, perhaps with a textual context, or a record with a few fields, what is the most likely target for mapping the string onto an individual entity or semantic class? A variant of this problem is to test two strings or records as to whether they are duplicates [97].

There is extensive literature on entity resolution. Typically, the first step is to define a similarity measure between strings/records in context and possible target entities (or among strings/records for de-duplication). For strings and entity names themselves, edit-distance-based measures such as the Jaro-Winkler distance are popular. Such measures can operate on flexible notions of tokens that are formed by pre-processing (e.g., stemmed words, PoS-tagged words, N-grams, etc.), and they can be combined and augmented with type-specific distances for numeric or categorical fields (e.g., spatial distances for cities or zip codes) (see, e.g., [42, 41, 83] and references given there). Machine learning techniques may be employed to consider the joint disambiguation of multiple entities [119]. For example, for detecting if two bibliographic entries with different spellings, abbreviations, and typos in authors and venues are the same or not, there is a mutual reinforcement effect between matching the different authors, the venues, etc. If you know that two authors are most likely the same, this affects the likelihood that two co-authors are the same as well. Of course, scalability is a key issue here, especially for the de-duplication setting [15, 33, 83]. State-of-the-art approaches work reasonably well, but there is considerable room for improvement. The problem of entity disambiguation remains a challenge.

Interestingly, the availability of large knowledge bases that know typed entities and many facts about them can boost the effectiveness of entity disambiguation. When we see a string (in context) on a Web page, text document, or database record, we can match it against all strings known in the knowledge base as synonyms of some entities. For example, the knowledge base would know both Michael Jeffrey Jordan and Michael I. Jordan, and further facts about them. If the string context mentions words like Chicago Bulls, center, or champion, we can perform a similar comparison against the words in the types and facts of Michael Jeffrey Jordan and would encounter high similarity. Thus, we would map this occurrence of “Michael Jordan” to the NBA player Michael Jeffrey Jordan. On the other hand, if the string context contains words like Berkeley, machine learning, or computer science, then Michael I. Jordan would be the better target. This is a fairly simple but amazingly effective heuristics, leveraging the situation that we already have a rich knowledge base with plenty of well-structured infor-

mation [45, 129]. It can be further strengthened in its accuracy, by constructing string contexts from multiple documents found on the Web [34].

Collecting entity synonyms. The last point raises the question: how do we avoid the scalability problem of having to compare a string (in context) against a huge number of possible target entities? And how do we identify the string itself as a worthwhile candidate for entity resolution? For the latter, we could run PoS tagging and ML techniques such as CRFs (see, e.g., [114]) for segmenting and labeling. But if the knowledge base is rich enough in terms of entity synonyms (including nicknames, abbreviations, and perhaps even common typos), it would be easier and way more efficient to query the knowledge base for matching strings. To build a fairly comprehensive collection of synonyms, YAGO has systematically harvested the Wikipedia redirection and disambiguation pages, and also considers strings in the ample href anchors that are used to cross-link articles. Of course, Wikipedia is not and will never be fully complete, neither in terms of synonyms nor in terms of interesting entities. But for many domains, explicit dictionaries are available, e.g., *imdb.com* for movies, *librarything.com* for books, *musicbrainz.org* for pop music, *geonames.org* for geographic places, DBLP for computer scientists, MeSH (Medical Subject Headings) and UMLS (Unified Medical Language System) and for biomedical entities, and so on. In addition, and most important for the rapidly changing world of product names, companies, and business people, [34, 35] have developed efficient techniques for mining news and Web sources to discover entity synonyms.

3.2 Problems and Challenges

Wikipedia categories reloaded. Since the early work on YAGO and WikiTaxonomy the Wikipedia category system has grown much larger and also become somewhat cleaner. There is definitely large room for improvement, especially regarding the *recall* in distilling classes and *subClassOf* relations. One interesting asset for re-addressing this issue is the large number of interwiki links across the Wikipedia editions in different languages. Each edition comes with its own category system. Semantically they should largely overlap, but they are independently crafted by different sub-communities. This should offer a fertile ground for better harvesting of taxonomic relations.

Tags and topics. While Wikipedia is the focal point of knowledge harvesting, there are plenty of other sources for mining taxonomic relations. These include “social tags” from online communities such as *del.icio.us*, *citeulike.org*, *librarything.com*, or *flickr.com*, and also tags assigned to blog postings and news articles. In addition, informally compiled directories such as *dmoz.org* are potentially valuable, although their directory structure is not based on taxonomic relations.

Long tail of entities. Going beyond the individual entities that are featured as Wikipedia articles would be a worthwhile goal. Even Wikipedia articles mention many people, organizations, and events that do not correspond to articles themselves. For example, can we systematically compile the spouses and children of all persons known to YAGO, from articles and also biographies and other Web sources? How can we build a machinery that automatically identifies new entities on the Web as they appear and become relevant (e.g., in the news)? Superficial approaches with high recall but low precision are easily conceivable, but high-precision methods that achieve near-human quality are a challenge. Even accurately determining all current students of all DBLP authors seems impossible today.

Robust disambiguation. The correct mapping of surface strings onto unique entities remains a very difficult problem. Heuristics go

a long way, and the proliferation of explicit knowledge assets is extremely helpful. However, a principled solution with very high accuracy is still missing. Perhaps, even a better theory is needed, for example, to characterize upper bounds for the best possible disambiguation given a limited amount of input context. Obviously, a single-keyword query such as “Jordan” can never be properly disambiguated (other than by guessing) if nothing else is known about the user and her situational context or long-term history.

4. RELATIONSHIPS

4.1 State of the Art

Equipped with the knowledge about individual entities and their types, we now aim to gather and clean facts about entities. Here we consider instances of binary relations. Note that with typed entities, these relations would have a type signature as well. For example, we could be interested in: $birthplace \subseteq \text{Person} \times \text{City}$, $marriedTo \subseteq \text{Person} \times \text{Person}$, $graduatedAt \subseteq \text{Person} \times \text{University}$, $hasAdvisor \subseteq \text{Scientist} \times \text{Scientist}$, $headquarteredIn \subseteq \text{Company} \times \text{City}$, $isCEOof \subseteq \text{Person} \times \text{Company}$, $playsInstrument \subseteq \text{Musician} \times \text{Instrument}$, and many more. We disregard ternary and higher-arity relationships, by assuming that there is typically a binary base fact to which additional facts can refer. For example, the year of graduation could be another relation $graduatedAtYear \subseteq graduatedAtFact \times \text{Year}$. This may look like a higher-order representation, but by reifying the base facts it can be cast into standard first-order logic.

The methods for harvesting relational facts pursue different paradigms: 1) rule-based with declarative querying as a key asset, 2) pattern-based drawing on NLP techniques and statistics, and 3) learning-based with joint inferencing by coupled learners and logical reasoning about hypotheses. Obviously, the rule-based paradigm is best suited for semistructured inputs such as Web tables, the pattern-based paradigm is needed for natural-language inputs, and the learning/reasoning-oriented paradigm aims to combine the best of both worlds. In the following we discuss prior and ongoing work in these three research avenues.

4.1.1 Rule/Query-based Methods

Wrappers and Wrapper Induction. From a DB perspective, the obvious idea is to exploit regularities in the structure of Web sources. In many cases, Web pages are actually generated from a database-backed content management system. Then it is possible to construct or automatically infer *wrappers* for fact extraction from HTML headings, tables, lists, form fields, and other semistructured elements. To this end, powerful languages for extraction scripts – typically centered around regular expressions over DOM trees – have been developed, and methods for learning structure from examples have been successfully applied (see, e.g., [9, 28, 85, 86, 112]). The latter is also known as *wrapper induction*. Some approaches employed ML techniques like HMMs and classifiers, but the general rationale has been to arrive at a set of good extraction rules that could be applied in a deterministic manner. Early prototype systems of this kind included Rapiere [28] and the W4F toolkit [112]; more recent systems that are being pursued further include Lixto [14, 30, 70], RoadRunner [43, 44], and SEAL [135].

Declarative extraction. More recent work on rule-based fact gathering is based on DB-style *declarative IE* (see [50] for several overview articles). [48, 108, 117] have shown how to combine query processing for extraction and consistency-centered inferencing into a unified framework. In the Cimple framework [117], non-declarative program parts (e.g., for text-pattern analysis) can be encapsulated into declarative programs in the XLog language. Consistency constraints are lifted into first-order logical rules, which can

then be checked in a unified way against both existing facts and candidate facts derived during the extraction process. SystemT [84, 93, 108] has developed a declarative language, coined AQL, for fact extraction tasks, along with an algebra and query rewriting rules.

A very nice showcase is the (at least largely) automated construction and maintenance of the *DBLife* community portal (dblife.cs.wisc.edu), which is based on the *Cimple* tool suite [48, 49]. DBLife features automatically compiled “super-homepages” of researchers with bibliographic data as well as facts about community services (PC work, etc.), colloquium lectures given at important institutions, and much more. For gathering and reconciling these facts, Cimple provides a suite of DB-style extractors based on pattern matching and dictionary lookups. These extractors are combined into execution plans, and periodically applied to a carefully selected set of relevant Web sources. The latter include prominent sites like DBLP and the dbworld messaging archive, but also important conference and university pages that are determined semi-automatically.

Efficiency and optimization. While declarative IE is a powerful paradigm, it gives rise to major issues for indexing and query processing. [1, 2] discuss scalability issues in IE. The QXtract system [2] aims to filter relevant from irrelevant documents sources for a given set of queries. The optimization framework by [76, 78, 79] aims to generate execution plans for extraction tasks and text-mining workflows. This entails trading off the costs of crawling documents vs. gathering relevant sources by appropriately generated queries. Although text-centric, this framework employs many ingredients from DB query optimization: selectivity estimation, plan generation, etc. [36, 37] present indexing strategies over evolving data. [39] provide SQL-like, structured queries directly over unstructured or loosely structured Wikipedia articles. [93, 108] have shown how to use algebraic representations for query optimization: pushing down expensive extraction operators and ordering join conditions (matching predicates) for lower-cost execution plans. [92] investigates efficient search techniques for Medline articles in the biomedical domain. Also in the medical domain, recent projects such as AliBaba [100] address efficiency aspects by selecting the most suitable patterns to be used for IE.

Wikipedia, again. Rule-based fact extraction has also been customized to Wikipedia as a knowledge source, more specifically to exploiting the great asset provided by infoboxes. Infoboxes are collections of attribute-value pairs. They are often based on templates and then reused for important types of entities such as countries, companies, scientists, music bands, sports teams, etc. For example, the infobox for Nicolas Sarkozy gives us data such as $birth_date = 28\ January\ 1955$, $birth_place = Paris$, $spouse = Carla\ Bruni$, $occupation = lawyer$, and $alma_mater = University\ of\ Paris\ X: Nanterre$. DBpedia [12] has pioneered the massive extraction of infobox facts. It uses simple, recall-oriented techniques and essentially places all attribute-value pairs into its knowledge base as they are. The values do not necessarily correspond to known (and typed) entities; values as well as attribute names are of mixed quality caused by non-sensical names (recall “*filler3*”) and non-unique naming. YAGO [124], on the other hand, uses a suite of carefully designed rules for frequently used infobox attributes to extract and normalize the corresponding values. For example, the *spouse* attribute is mapped to the *marriedTo* relation, and the extracted fact then is *Nicolas Sarkozy marriedTo Carla Bruni*. YAGO did not attempt to extract *all* infobox attributes, as their “long tail” has a lot of naming diversity and noise. The Kylin/KOG project [138, 139, 140] is a more recent attempt at universally harvesting infoboxes, with careful cleaning based on advanced ML techniques.

Type checking. As YAGO has all entities typed by their assignment to one or more classes, binary relations have a type signature as well. For example, the *isCEOof* relation can be specified to have a domain *BusinessPerson* or simply *Person* and a range *Company* (where we denote relations as powerset functions). When the extractor produces a candidate fact like *Nicolas Sarkozy isCEOof France*, we can reject it because the second argument, *France*, is not a company. Similarly, the hypothesis *Nicolas Sarkozy married-To Élysée Palace* which may, perhaps, be incorrectly inferred from sentences such as “Nicolas Sarkozy loves his work with the Élysée Palace” (or from an incorrect interpretation of the infobox attribute “Residence”), is falsified by the type invariant that marriages are between persons. Type checking is a powerful building block for high-precision harvesting of relational facts. It is enabled by the methods of Section 3 for gathering individual entities and carefully organizing them into semantic classes with a rigorous *subclassOf* hierarchy.

4.1.2 Pattern-based Methods

Hand-crafted patterns. Using textual patterns for fact extraction from natural-language documents has a long history in the NLP and AI communities, dating back to the works by Hindle [74] and Hearst [73]. [74] proposed unsupervised clustering techniques to identify relationships among entities, based on the observation that natural language has restrictions on which noun phrases can occur as subject or object of a given verbal phrase. For example, *wine* may be *drunk* or *produced*, but not *eaten* or *driven*. Thus, noun phrases can be clustered based on their appearance in similar verbal phrases. Hearst patterns [73] are PoS-enriched regular expressions (so-called *lexico-syntactic* patterns) which aim to identify instances of predefined relationship types from free text. For example, for the *instanceOf* relation we can automatically determine instances from noun phrases around a syntactic pattern like $\langle NP_0 \text{ such as } \{NP_1, NP_2 \dots (\text{and/or})\} NP_n \rangle$ (where NP is the PoS tag for proper nouns). Hearst patterns have later been defined also for other taxonomic relations [77], including *partOf* [18] and also causal relations [69]. Hearst patterns yield relatively high precision, but typically suffer from low recall due to the sparseness of the exact patterns. In any case, the patterns are hand-crafted; for arbitrary target relations (e.g., *marriedTo* or *hasAdvisor*) it would be difficult to come up with expressive yet accurate patterns.

Pattern-fact duality. The early work on hand-crafted patterns led to the goal of automatically constructing characteristic patterns for a given relation of interest. The seminal work by Brin [22] was centered around the following observation on the *duality of facts and patterns*: if we knew enough facts for a relation (e.g., instances of married couples) we could automatically find textual patterns and distill the best ones, and if we knew good patterns, we could automatically find more facts. Initially, this mutually recursive gathering process may look like a daunting ML task, but extensive hand-labeling for training would usually be out of the question anyway. Instead, [22] suggested the following *almost-unsupervised* method, which has later become the prevalent approach. The fact harvesting starts with a small set of *seed facts* for one or more relations of interest, then automatically finds markup, textual, or linguistic *patterns* in the underlying sources as indicators of facts, and finally uses these patterns to identify new *fact candidates* as further hypotheses to populate the relations in the knowledge base. For example, for collecting facts about the Alma Mater and doctoral advisor of researchers, one could start with seeds such as *AlmaMater(Jim Gray, UC Berkeley)*, *AlmaMater(Hector Garcia-Molina, Stanford)*, *Advisor(Jim Gray, Mike Harrison)*, and *Advisor(Hector Garcia-Molina, Gio Wiederhold)*. We could then find text patterns such as “*x* gra-

duated at *u*”, “*x* and his advisor *y*”, and “professor *y* and his student *x*” (with placeholders *x*, *u*, *y* for the involved named entities), which in turn could lead to discovering new facts such as *AlmaMater(Susan Davidson, Princeton)* and *Advisor(Susan Davidson, Hector Garcia-Molina)*. This process of gathering and generating facts and patterns is then iterated.

At each iteration, some form of *statistical assessment* is needed to quantify the indicative strengths of patterns and the confidence in fact candidates and to prune noisy candidates accordingly. For example, we could use a search engine or sample a large Web corpus to compute the (point-wise) *mutual information* (MI) between the two entities in a fact or between a pattern and a fact that it supports. This is a frequency-based measure for the relative entropy (Kullback-Leibler divergence) of two observations (e.g., the two entities in a fact) co-occurring together vs. being independently observed (i.e., co-occurring accidentally at random). Only the facts and patterns with MI measures above some threshold would be kept for the knowledge base and the next iteration of the harvesting process. Additional plausibility tests can be employed based on logical consistency constraints, to reduce the false-positive rate. For example, if researcher *x* graduated from university *u* under the supervision of *y*, then *y* would have to be a professor or lecture with a position at or other connection to *u*. We will come back to this theme further below.

Advanced pattern generation. The iterative process outlined above is powerful, but difficult to tune (regarding thresholds, weighting parameters, etc.) and susceptible to drifting away from its target. While high recall is easily possible, the precision would often be unacceptable. This led to a series of improvements in a variety of projects and tool developments, most notably, Snowball [3], Semagix/SWETO [5, 118], KnowItAll [59], Text2Onto [21, 40], LEILA [122], TextRunner [13, 142], SEAL [134], and the work by [23] and [141] (and others). Snowball, KnowItAll, and Text2Onto improved the statistical assessment of fact candidates and patterns in a variety of ways, regarding robustness (lower false-positive rate while still retaining high recall), expressiveness (e.g., by adding PoS tagging and other NLP-based features), and efficiency (lower-cost estimates for MI and other measures). Semagix/SWETO integrated heuristics for entity disambiguation into the fact extraction process. SEAL combined the seed-based harvesting with graph-proximity-oriented assessment. [23] proposed the use of lexical dependency parsing for a richer representation of the sentences before identifying patterns; pattern candidates would then be the word sequences on the shortest paths between the two entities that form the fact hypothesis. [141] extended these approaches to extracting higher-arity relations and complex events. LEILA [122] used dependency-parsing-based features for boosted precision, and also extended the bootstrapping technique by incorporating both positive and *negative seeds*. Negative seeds are entity pairs which are known to *not* belong to the relation of interest; these may include generalized patterns such as regular expressions over tokens. The negative seeds help to discover spurious patterns even if they receive support from the positive seeds. For example, for the *isCapitalOf* relation, seeds like *(Paris, France)* and *(Berlin, Germany)* can easily lead to the poor pattern “*x* is the largest city of *y*”; negative seeds like *(Sydney, Australia)* or *(Rio de Janeiro, Brazil)* can counter this effect. TextRunner extended the pattern-fact bootstrapping paradigm to Open IE, where the harvesting is not focused on a particular relation but considers all relationships expressed in verbal phrases.

Semistructured inputs. Pattern-based extraction was designed for natural-language input, but is equally suited for semi-structured inputs such as Web tables or lists. For example, patterns such as

“birth date: y ” can be learned and applied if the left-hand argument (person x) can be inferred from the context (e.g., the title of a homepage or Wikipedia article). DOM-tree-based patterns that combine, for example, a column name in a table header with the value in a table cell, can also be cast into the fact-pattern duality framework. However, the output of such an approach may need substantial postprocessing (e.g., clustering, de-duplicating, plausibility testing, etc.). [25, 56] present recent approaches for tapping on HTML tables and lists at Web scale.

Another form of structured-inputs situation is to start with several formalized knowledge sources and obtaining better knowledge bases by integrating them. This is also known as *ontology merging*. [90] has developed mapping methods for the automatic alignment of multiple ontological sources. [131] combined logical reasoning and statistical arguments for merging knowledge from multiple, heterogeneous ontologies. [89] devised ways of (semi-) automatically extracting smaller but high-quality, domain-specific ontologies from specific Web sources through interaction with a human ontology engineer. An overview of approaches for involving humans for quality assurance or active learning, including “wisdom of the crowd” games, is given in [52].

4.1.3 Learning/Reasoning-based Methods

To reconcile the high recall provided by pattern-based harvesting with the high precision of rule-based methods, pattern-based methods can be augmented by an additional form of *joint/constrained reasoning* on the set of fact hypotheses gathered by a recall-oriented method. Consider, for example, a situation where we have compiled several hypotheses about facts of the *marriedTo* relation: (*Carla, Nicolas*), (*Carla, Mick*), (*Carla, Benjamin*), (*Bianca, Mick*), (*Chantal, Nicolas*). If we independently accept some of these hypotheses as new facts for the knowledge base, we may end up with the same person having multiple spouses (at the same time – assume we are looking at a snapshot in time). If, on the other hand, we consider the joint probability distribution for these hypotheses being true or false, we have a better perspective for accepting the truly valid ones. Moreover and most importantly, if we add a constraint that *marriedTo* is an injective function (for any given timepoint), we may be able to eliminate the false hypotheses by means of constraint violation. This form of consistency awareness is in fact a DB paradigm added to the more ML/AI-oriented harvesting methods considered so far.

Statistical Relational Learning. The field of statistical relational learning (SRL) [67] has gained strong interest in both the AI and DB communities. Within the SRL family, Markov Logic Networks (MLN) [53, 110] are probably the most versatile approach in combining first-order logic rules and probabilistic graphical models; the *Alchemy* open-source software implements a wide variety of MLN-based algorithms [4]. The Markov Logic framework works by *grounding* rules against base facts and newly extracted candidate facts: substituting constants for the variables in the rules so as to produce a set of propositional-logic clauses without any variables. Now the literals (concrete fact hypotheses) in the clauses are interpreted as binary random variables, with a probabilistic dependency between two literals if they appear in the same clause. Thus, the interconnection of clauses determines the complexity of reasoning about this setting. This probabilistic structure forms a Markov Random Field (MRF); we can view this as the machine-language representation of the more elegant MLN-level representation. The goal of the MLN solver is to compute the joint probability distribution of all variables in the MRF, or to determine the most likely joint assignment of truth values. Like in most probabilistic graphical models, inference in MLNs – actually the underlying MRFs – is

NP-hard. This is why efficient approximations and techniques for Markov Chain Monte Carlo (MCMC) sampling [106, 109] need to be employed.

Kylin/KOG [138, 139, 140] is an interesting application of MLNs and a suite of other learning techniques. It aims to infer “missing infobox values” in Wikipedia. Consider an entity of a given type (e.g., scientist) for which many Wikipedia entities have a rich infobox (e.g., with values for the *Alma Mater*), but the given entity is lacking one or more attributes in its infobox or does not have an infobox at all (which is indeed the case for many entities “in the long tail”). The existing infoboxes for this type (as well as other assets such as the YAGO taxonomy) are used for learning a model by which the entity type and missing attribute values can be inferred.

Constraints Conditional Models (CCMs) [32, 111] are another way of combining declarative and probabilistic models and have been applied to IE tasks such as entity disambiguation and semantic role labeling. In the ReadTheWeb project [29], semi-supervised learning ensembles have been combined with constraints for extracting entities and facts from a huge Web corpus. Efficiency is a major issue for these methods: CCMs are mapped into integer linear programs, ReadTheWeb uses massive parallelism on a large cluster (but does not report any figures on resource consumption and run-time).

Leveraging existing knowledge. Parallel work that has found new ways of combining pattern-based harvesting with consistency reasoning is the *SOFIE* methodology [121, 125], which was developed to enable automatic growth of YAGO while retaining the high level of near-human quality. SOFIE maps all ingredients – known facts from the knowledge base, new fact hypotheses, patterns, constraints, and possible entity disambiguations – into a set of weighted clauses, where the weights are derived from the automatically gathered statistical evidence. This representation is also grounded, just like in the MLN-based approaches. In contrast to MLNs, however, the goal of the reasoning is not to infer a joint probability distribution, but SOFIE settles for the somewhat simpler goal of computing truth values such that the total weight of satisfied clauses is maximized. This is also NP-hard, but there are many good approximation algorithms, and SOFIE customized various elements from the prior work on MaxSat solvers to the specific structure of clauses in its knowledge harvesting and assessment setting. This unified reasoning includes entity disambiguation, provides very high precision (while restricting recall), and is reasonably efficient: an overnight run on a standard PC can process thousands of rich Web pages (e.g., biographies or Wikipedia articles), with the grounding being the most expensive part.

SOFIE exploits YAGO and its rigorous entity typing for both accuracy and efficiency. As for accuracy, recall that all entities are typed by their assignment to one or more classes. Thus, binary relations have a type signature as well. This way, incorrect hypotheses such as *Jim Gray hasAlmaMater San Francisco* can be immediately falsified because San Francisco is a city and not a university, and thus violates the type condition. In the same vein, constraints about types and relations are a very effective means of hardening the output of SOFIE. As for efficiency, the typing has two highly beneficial effects. First, many type-incorrect hypotheses can be pruned early before entering the reasoning phase. Second, in grounding the first-order constraints, we need to consider only constants (entities from the knowledge base) that match the required types in the constraints. This reduces the size of the clauses set by a large margin, and speeds up the subsequent reasoning as well. While these techniques are not rocket science, they clearly demonstrate the need for clever engineering and the great asset than an existing knowledge core, such as YAGO, provides to further growing a fact collection.

This theme – use knowledge to better acquire more knowledge – is an exciting direction to be explored further.

Open Information Extraction. Recent work by [24, 26, 27] addresses the goal of *open information extraction* (Open IE). The TextRunner system [13, 142] aims at extracting all meaningful relations from Web pages (coined “*assertions*”), rather than a predefined set of canonical relations. Here, entities are not yet necessarily disambiguated (i.e., the same extracted name may refer to different real-world entities), and relations are not canonicalized: all verbal phrases found in natural language sentences may constitute a valid relation type between two or more entities. These two relaxations make it very difficult to apply any form of consistency reasoning on the extracted data.

4.2 Problems and Challenges

Reconciling high precision and high recall. The biggest challenge is to achieve both high precision and high recall. Achieving one while neglecting the other is easy, but automatically constructing knowledge bases with near-human quality requires both. To this end, combining pattern-based methods (for high recall) and reasoning about candidate facts as hypotheses (for high precision) in a declarative manner seems to be the most promising avenue. A great advantage of a declarative approach is the ability to treat constraints (and other logical dependencies) as first-order rules and apply them to millions of candidate facts “at once”. Today’s methods work by grounding rules and combining the grounded model with both seed facts and newly found candidate facts. It is conceivable, though, and may be very worthwhile to pursue that constraints are dealt with in their lifted first-order form, without any tedious grounding. However, this would probably entail the need for adopting full-fledged theorem proving methods from computational logics.

Extending and optimizing declarative IE. The DB-oriented approaches of declarative extraction languages have the advantage of enabling a suite of query optimization techniques. However, the current generation of these approaches is still primarily geared for semistructured inputs such as Web tables. It is not clear how to generalize to arbitrary domains and input types, including natural language. For example, how would one incorporate pattern-based and learning-based methods into a declarative extraction language, with full capabilities for execution-plan rewriting and optimization?

Types and constraints. Typed entities and type signatures for relations can drastically reduce the number of grounded instances. Typing and consistency constraints are also powerful assets for improving precision. In the MLN-based and Max-Sat-based work, constraints are typically treated in a *soft* manner: they can be violated (by some instances in the grounded model), but this incurs a penalty. In some cases, however, we may want to enforce *hard* constraints. How to deal with the co-existence of both soft and hard constraints is an open issue. Overemphasizing hard constraints may result in empty results, while soft constraints could dilute the outcome of the reasoning. A unified reasoning framework thus requires a statistically quantified impact of soft constraints, which may be violated by a certain amount of instances.

Inconsistencies. Should we aim to keep the knowledge base consistent at all times and perform eager cleaning, or would it have advantages to keep diverse, potentially inconsistent hypotheses as first-class citizens and rather perform uncertainty-aware reasoning at query time? The latter faces on-line efficiency challenges in resolving inconsistencies at query time or whenever required by an application task, with interactive response times.

Scale and dynamics. For a universal knowledge base, one would need to process a very large set of Web sources, potentially on a daily basis, as the world progresses rapidly. Even when looking only at high-quality sources such as news articles and scientific publications, the rate at which online-available knowledge grows poses major challenges to any harvesting approach. Even the rate of major edits of Wikipedia articles presents a challenge. Imagine computing all spouses and their validity periods of all people mentioned in Wikipedia – in one night, with > 90 percent precision and > 90 percent recall. This would be quite a challenging benchmark. Perhaps, knowledge harvesting needs to be adapted to a cloud-style or distributed infrastructure with massively parallel computation. Some of the steps in knowledge harvesting are embarrassingly parallel (e.g., pre-processing incoming Web sources by NLP tools), but others would be difficult to parallelize (e.g., consistency reasoning).

Open IE and generic reasoning. In the long-term, targeted IE with a fixed set of relations will be insufficient, and open IE for arbitrary, previously unknown, relation types would be the ultimate goal. How can we make the few existing approaches to open IE more robust for rigorous knowledge base building (with canonicalized entity and relation names), or at least for extending an existing knowledge base? What is the role of consistency reasoning in such an endeavor? Non-canonical facts (with diverse naming) make it difficult to apply the currently used reasoning approaches. Perhaps, new combinations of reasoning with semi-supervised ML methods may hold the answer?

5. TEMPORAL KNOWLEDGE

5.1 State of the Art

So far we have simplified our knowledge-harvesting setting by assuming that facts are time-invariant. This is appropriate for some relation types, for example, for finding birth dates of famous people, but inappropriate for evolving facts, e.g., presidents of countries or CEOs of companies. In fact, time-dependent relations seem to be far more common than time-invariant ones. For example, finding all spouses of famous people, current and former ones, involves understanding temporal relations. Extracting the validity time of facts involves detecting explicit temporal expressions such as dates as well as implicit expressions in the form of adverbial phrases such as “last Monday”, “next week”, or “years ago”. Moreover, one often has to deal with incomplete time information (e.g., the begin of someone holding a political office but no end-of-term date given, although the person may meanwhile be dead), and with different time resolutions (e.g., only the year and month for the begin of the term, but the exact date for related events). In addition to the complexity of extracting temporal knowledge, this also entails difficult issues of appropriately reasoning about interrelated time points or intervals. For example, the constraint that each person has at most one legal spouse now becomes a more complex condition that the validity intervals of the *marriedTo* instances for the same person must be non-overlapping.

Initial work on these issues includes [7, 17, 101, 136, 143]. For illustration of conceivable approaches, we briefly outline our preliminary work towards a *Timely-YAGO* (T-YAGO) knowledge base [136]. We first use - time-ignorant - fact extraction techniques to collect base facts and sentences (or semistructured elements such as lists, tables, infoboxes) in which they are expressed. Then we analyze these sentences and their contexts for additional information about the temporal scope of the facts. Temporal expressions can be explicit expressions like dates or implicit expressions like adverbial phrases. The former can be extracted by regular expression

matching, the latter require deeper NLP and/or a good dictionary of temporal expressions [133]. For both it is often necessary to a) validate that they actually refer to the considered fact (and not to another aspect of the same sentence) and b) determine the exact denotation that connects the fact and the temporal expression. For example, an expression may denote the beginning of an interval during which the fact holds, its end, both, or a relative timepoint or interval.

To cope with the variety of temporal expressions, a unified representation is helpful. Explicit expressions should have (earliest, latest) bounds for timepoints and the begin and end of intervals [143]. This is convenient for capturing different resolutions (e.g., “July 2009” vs. “2009”), and checking, cleaning, and refining temporal information. Alternatively, one could represent the uncertain temporal scope of a fact by means of (time-range-bucketized) histograms [115] or other kinds of probabilistic models (e.g., a Gaussian for the uncertain timepoint of an event). The DB and AI fields have strong track records on temporal models and may offer further alternatives for representing temporal knowledge (see, e.g., [10, 64]).

As T-YAGO is based on an RDF-style data model, we can capture only binary relations whereas temporal facts would rather appear to be higher-arity relations or even higher-order facts. We circumvent this issue by reifying base facts, giving them explicit identifiers, and then using these identifiers in additional facts about time scopes (as if the identifiers were constant literals in the underlying logical model). For example, for the ternary relation between Nicolas Sarkozy, the Grand Cross of the Légion d’Honneur, and 2007, we use the original time-agnostic fact as a primary fact with identifier #1. For temporal facts valid at a time point, we use the relation *onDate* to describe the validity time. Then we represent the temporal property of the primary fact as: #1 *onDate* 2007. For temporal facts that are valid during a time period, we use two relations to represent the interval: the relation *since* for the begin time point, and the relation *until* for the end time point. For example, the fact that Nicolas Sarkozy was Mayor of Neuilly-sur-Seine from 1983 to 2002 is represented as #2: *Nicolas_Sarkozy mayorOf Neuilly-sur-Seine*, #3: #2 *since* 1983, and #4: #2 *until* 2002.

Sometimes it is impossible to extract accurate time-points, say the exact day, and sometimes we may know only the begin or the end of a fact’s validity interval but not both. For these situations, we use the above mentioned representation for time-points with the earliest and latest possible time to constrain the range of the true time point. For example, if we know that Nicolas Sarkozy was elected for Mayor of Neuilly-sur-Seine on April 29, 1983 and resigned from this office on May, 2002 we would add the temporal facts #3: #2 *since* [29-April-1983, 29-April-1983] and #4: #2 *until* [1-May-2002, 31-May-2002]. If we later figure out that he resigned from this office exactly on May 7, 2002, we are able to refine the *until* relation for #2 into #4: #2 *until* [7-May-2002, 7-May-2002].

For querying a time-aware knowledge base, languages like SPARQL need extensions, too. Reification and fact identifiers or fact-identifier variables can be added to the concept of triple patterns, and additional temporal predicates such as *during*, *overlap*, *inYear*, etc. are needed. These issues and also the ranking of temporal search results are being investigated in the exploratory work of [17, 101, 107, 127, 136]; aggregating temporal facts on a per-entity basis and visualizing them as timelines is addressed in [7] and the Timeline Project (see www.simile-widgets.org/timeline).

5.2 Problems and Challenges

This aspect of temporal knowledge harvesting is relatively new and the ongoing work along these lines is fairly preliminary. Thus, there are plenty of research opportunities.

Incomplete and uncertain temporal scopes. In knowledge harvesting, incompleteness and uncertainty are intrinsic features of temporal facts. How to represent them in an expressive yet easy-to-manage way is a widely open area. We need to capture contradictory information about the begin, end, duration, and perhaps even relative ordering of relational instances (e.g., a marriage or the term of a political office), and we need to cope with different time resolutions and missing pieces of information. The representation must support refinements as more evidence is collected from Web sources. While prior work on temporal modeling (mostly in AI and DB theory) is highly relevant, we believe that the requirements for the specific purpose of knowledge harvesting are sufficiently different to justify and inspire a fresh look.

Consistency reasoning. For assessing and cleaning temporal hypotheses in a knowledge base, consistency reasoning may play a key role. For example, marriages of the same person must not overlap in time. Other constraints may refer to the relative ordering of facts: if *X* graduated with doctoral advisor *Y*, then *Y* herself/himself must have graduated before *X*. When reasoning about the validity of facts in the presence of such constraints, both base facts and facts about temporal scopes need to be considered together. For example, if we see strong evidence that *X* graduated before *Y*, perhaps this is correct and *Y* was not the true advisor of *X*. We believe that reasoning methods along these lines require a carefully designed integration of logical and statistical elements.

Narrative information. Many biographies, which are among the richest sources for temporal knowledge, have a narrative structure. They report events in chronological order, without necessarily giving explicit dates for each and every fact. The issue here is how to exploit these relative orderings for knowledge extraction. This may require very advanced NLP methods, but should also tie in with the temporal consistency reasoning pointed out above.

Temporal querying. Although we sketched some possible approaches to querying and exploring temporal knowledge, the principled design of a query language and the development of efficient query processing algorithms are widely open. Obviously, this can build on the extensive prior work on temporal databases, but it also needs to consider the very different setting now: no schema, need for flexible search (incl. combined fact-keyword predicates), different confidence in different pieces of knowledge, ranking as a first-class citizen.

6. REFERENCES

- [1] E. Agichtein. Scaling information extraction to large document collections. *IEEE Data Eng. Bull.*, 28(4), 2005.
- [2] E. Agichtein, L. Gravano. Querying text databases for efficient information extraction. *ICDE*, 2003.
- [3] E. Agichtein, L. Gravano, J. Pavel, V. Sokolova, A. Voskoboinik. Snowball: a prototype system for extracting relations from large text collections. *SIGMOD*, 2001.
- [4] Alchemy—Open-Source AI: alchemy.cs.washington.edu
- [5] B. Aleman-Meza, C. Halaschek, A. Sheth, I. B. Arpinar, G. Sannapareddy. SWETO: Large-scale semantic web test-bed. *SEKE: Workshop on Ontology in Action*, 2004.
- [6] F. Alkhateeb, J.-F. Baget, J. Euzenat. Extending SPARQL with regular expression patterns (for querying RDF). *Web Semant.*, 7(2), 2009.
- [7] O. Alonso, M. Gertz, R. A. Baeza-Yates. Clustering and exploring search results using timeline constructions. *CIKM*, 2009.
- [8] K. Anyanwu, A. Maduko, A. Sheth. SPARQ2L: towards support for subgraph extraction queries in RDF databases. *WWW*, 2007.
- [9] A. Arasu, H. Garcia-Molina. Extracting structured data from web pages. *SIGMOD*, 2003.
- [10] A. Artale, E. Franconi. Foundations of temporal conceptual data models. *Conceptual Modeling: Foundations and Applications*, 2009.
- [11] N. Ashish, C. A. Knoblock. Semi-automatic wrapper generation for internet information sources. *COOPIS*, 1997.

- [12] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives. DBpedia: A nucleus for a web of open data. *ISWC*, 2007.
- [13] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni. Open information extraction from the web. *IJCAI*, 2007.
- [14] R. Baumgartner, S. Flesca, G. Gottlob. Visual web information extraction with Lixto. *VLDB*, 2001.
- [15] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, J. Widom. Swoosh: a generic approach to entity resolution. *VLDB J.*, 18(1), 2009.
- [16] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, J. Widom. Databases with uncertainty and lineage. *VLDB J.*, 17(2), 2008.
- [17] K. Berberich, S. Bedathur, O. Alonso, G. Weikum. A language modeling approach for temporal information needs. *ECIR*, 2010.
- [18] M. Berland, E. Charniak. Finding parts in very large corpora. *ACL*, 1999.
- [19] T. Berners-Lee, J. Hendler, O. Lassila. The semantic web. *Scientific American*, 2001.
- [20] P. A. Bernstein, L. M. Haas. Information integration in the enterprise. *Commun. ACM*, 51(9), 2008.
- [21] S. Blohm, P. Cimiano. Using the web to reduce data sparseness in pattern-based information extraction. *PKDD*, 2007.
- [22] S. Brin. Extracting patterns and relations from the World Wide Web. *WebDB*, 1998.
- [23] R. Bunescu, R. Mooney. Extracting relations from text: From word sequences to dependency paths. *Text Mining & Natural Language Processing*, 2007.
- [24] M. J. Cafarella. Extracting and querying a comprehensive web database. *CIDR*, 2009.
- [25] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, Y. Zhang. WebTables: exploring the power of tables on the web. *PVLDB*, 1(1), 2008.
- [26] M. J. Cafarella, A. Y. Halevy, N. Khoussainova. Data integration for the relational web. *PVLDB*, 2(1):1090–1101, 2009.
- [27] M. J. Cafarella, J. Madhavan, A. Y. Halevy. Web-scale extraction of structured data. *SIGMOD Record*, 37(4), 2008.
- [28] M. E. Califf, R. J. Mooney. Relational learning of pattern-match rules for information extraction. *AAAI*, 1999.
- [29] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr., T. M. Mitchell. Coupled semi-supervised learning for information extraction. *WSDM*, 2010. <http://rtw.ml.cmu.edu/readtheweb.html>
- [30] J. Carme, M. Ceresna, O. Frölich, G. Gottlob, T. Hassan, M. Herzog, W. Holzinger, B. Krüpl. The Lixto project: exploring new frontiers of web data extraction. *BNCOD*, 2006.
- [31] S. Chakrabarti. Dynamic personalized PageRank in entity-relation graphs. *WWW*, 2007.
- [32] M.-W. Chang, L.-A. Ratniov, N. Rizzolo, D. Roth. Learning and inference with constraints. *AAAI*, 2008.
- [33] S. Chaudhuri, V. Ganti, R. Motwani. Robust identification of fuzzy duplicates. *ICDE*, 2005.
- [34] S. Chaudhuri, V. Ganti, D. Xin. Exploiting web search to generate synonyms for entities. *WWW*, 2009.
- [35] S. Chaudhuri, V. Ganti, D. Xin. Mining document collections to facilitate accurate approximate entity matching. *PVLDB*, 2(1), 2009.
- [36] F. Chen, A. Doan, J. Yang, R. Ramakrishnan. Efficient information extraction over evolving text data. *ICDE*, 2008.
- [37] F. Chen, B. J. Gao, A. Doan, J. Yang, R. Ramakrishnan. Optimizing complex extraction programs over evolving text data. *SIGMOD*, 2009.
- [38] T. Cheng, X. Yan, K. C.-C. Chang. EntityRank: searching entities directly and holistically. *VLDB*, 2007.
- [39] E. Chu, A. Baid, T. Chen, A. Doan, J. F. Naughton. A relational approach to incrementally extracting and querying structure in unstructured data. *VLDB*, 2007.
- [40] P. Cimiano, J. Völker. Text2Onto – a framework for ontology learning and data-driven change discovery. *NLDB*, 2005.
- [41] W. W. Cohen. A century of progress on information integration: a mid-term report. *WebDB*, 2005.
- [42] W. W. Cohen, P. Ravikumar, S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. *IJCAI*, 2003.
- [43] V. Crescenzi, G. Mecca. Automatic information extraction from large websites. *J. ACM*, 51(5), 2004.
- [44] V. Crescenzi, G. Mecca, P. Merialdo. RoadRunner: Towards automatic data extraction from large web sites. *VLDB*, 2001.
- [45] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. *EMNLP-CoNLL*, 2007.
- [46] H. Cunningham. An Introduction to Information Extraction, *Encyclopedia of Language and Linguistics (2nd Edition)*. Elsevier, 2005.
- [47] N. N. Dalvi, C. Ré, D. Suciu. Probabilistic databases: diamonds in the dirt. *Commun. ACM*, 52(7), 2009.
- [48] P. DeRose, W. Shen, F. Chen, A. Doan, R. Ramakrishnan. Building structured web community portals: A top-down, compositional, and incremental approach. *VLDB*, 2007.
- [49] P. DeRose, W. Shen, F. Chen, Y. Lee, D. Burdick, A. Doan, R. Ramakrishnan. DBLife: A community information management platform for the database research community. *CIDR*, 2007.
- [50] A. Doan, L. Gravano, R. Ramakrishnan, S. Vaithyanathan. (Eds.). Special issue on information extraction. *SIGMOD Record*, 37(4), 2008.
- [51] A. Doan, R. Ramakrishnan, F. Chen, P. DeRose, Y. Lee, R. McCann, M. Sayyadian, W. Shen. Community information management. *IEEE Data Eng. Bull.*, 29(1), 2006.
- [52] A. Doan, R. Ramakrishnan, A. Y. Halevy. Mass collaboration systems on the World Wide Web. *Comm. ACM*, 2010.
- [53] P. Domingos, D. Lowd. Markov Logic: An Interface Layer for Artificial Intelligence. *Morgan & Claypool*, 2009.
- [54] X. Dong, A. Y. Halevy, C. Yu. Data integration with uncertainty. *VLDB*, 2007.
- [55] S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, G. Weikum. Language-model-based ranking for queries on RDF-graphs. *CIKM*, 2009.
- [56] H. Elmelegy, J. Madhavan, A. Halevy. Harvesting relational tables from lists on the web. *PVLDB*, 2(1), 2009.
- [57] F. L. et al. Introducing meta-services for biomedical information extraction. *Genome Biology 9 Suppl. 2*, 2008.
- [58] O. Etzioni, M. Banko, M. J. Cafarella. Machine reading. *AAAI*, 2006.
- [59] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, A. Yates. Web-scale information extraction in KnowItAll. *WWW*, 2004.
- [60] O. Etzioni, M. J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1), 2005.
- [61] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1), 2005.
- [62] H. Fang, C. Zhai. Probabilistic models for expert finding. *ECIR*, 2007.
- [63] C. Fellbaum, editor. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, 1998.
- [64] M. Fisher, D. M. Gabbay, L. Vila (Eds.). Handbook of temporal reasoning in artificial intelligence. *Elsevier*, 2005.
- [65] M. J. Franklin, A. Y. Halevy, D. Maier. A first tutorial on dataspace. *PVLDB*, 1(2), 2008.
- [66] D. Freitag, A. McCallum. Information extraction with HMM structures learned by stochastic optimization. *AAAI/IAAI*, 2000.
- [67] L. Getoor, B. E. Taskar (Eds.). An Introduction to Statistical Relational Learning. *MIT Press*, 2007.
- [68] D. Gildea, D. Jurafsky. Automatic labeling of semantic roles. *Comput. Linguist.*, 28(3), 2002.
- [69] R. Girju, D. Moldovan. Text mining for causal relations. *FLAIRS*, 2002.
- [70] G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, S. Flesca. The Lixto data extraction project - back and forth between theory and practice. *PODS*, 2004.
- [71] A. Halevy, M. Franklin, D. Maier. Principles of dataspace systems. *PODS*, 2006.
- [72] O. Hartig, C. Bizer, J.-C. Freytag. Executing SPARQL queries over the web of linked data. *ISWC*, 2009.
- [73] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. *COLING*, 1992.
- [74] D. Hindle. Noun classification from predicate-argument structures. *ACL*, 1990.
- [75] V. Hristidis, H. Hwang, Y. Papakonstantinou. Authority-based keyword search in databases. *ACM Trans. Database Syst.*, 33(1), 2008.
- [76] P. G. Ipeirotis, E. Agichtein, P. Jain, L. Gravano. Towards a query optimizer for text-centric tasks. *ACM Trans. Database Syst.*, 32(4), 2007.
- [77] L. Iwanska, N. Mata, K. Kruger. Fully automatic acquisition of taxonomic knowledge from large corpora of texts: Limited syntax knowledge representation system based on natural language. *ISMIS*, 1999.
- [78] A. Jain, P. G. Ipeirotis. A quality-aware optimizer for information extraction. *ACM Trans. Database Syst.*, 34(1), 2009.
- [79] A. Jain, P. G. Ipeirotis, A. Doan, L. Gravano. Join optimization of information extraction output: Quality matters! *ICDE*, 2009.
- [80] D. Jurafsky, J. H. Martin. Speech and Language Processing (2nd Edition). *Prentice Hall*, 2008.
- [81] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum. NAGA: Searching and ranking knowledge. *ICDE*, 2008.
- [82] P. Kingsbury, M. Palmer. From Treebank to Propbank. *LREC*, 2002.
- [83] N. Koudas, S. Sarawagi, D. Srivastava. Record linkage: similarity measures and algorithms. *SIGMOD*, 2006. <http://queens.db.toronto.edu/~koudas/docs/aj.pdf>
- [84] R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, S. Vaithyanathan, H. Zhu. SystemT: a system for declarative information extraction. *SIGMOD Record*, 37(4), 2008.
- [85] N. Kushmerick. Wrapper induction: efficiency and expressiveness. *Artif. Intell.*, 118(1-2), 2000.
- [86] N. Kushmerick, D. S. Weld, R. Doorenbos. Wrapper induction for information extraction. *IJCAI*, 1997.

- [87] D. B. Lenat. CYC: a large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11), 1995.
- [88] D. B. Lenat, R. V. Guha. Building Large Knowledge-Based Systems; Representation and Inference in the CYC Project. *Addison-Wesley Longman Publishing Co., Inc.*, 1989.
- [89] A. Maedche, S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2), 2001.
- [90] A. Maedche, S. Staab. Measuring similarity between ontologies. *EKAW*, 2002.
- [91] C. Manning, H. Schütze. Foundations of Statistical Natural Language Processing. *MIT Press*, 1999.
- [92] J. D. Martin. Fast and furious text mining. *IEEE Data Eng. Bull.*, 28(4), 2005.
- [93] E. Michelakis, R. Krishnamurthy, P. J. Haas, S. Vaithyanathan. Uncertainty management in rule-based information extraction systems. *SIGMOD*, 2009.
- [94] Y. Miyao, T. Ohta, K. Masuda, Y. Tsuruoka, K. Yoshida, T. Ninomiya, J. Tsujii. Semantic retrieval for the accurate identification of relational concepts in massive textbases. *COLING-ACL*, 2006.
- [95] A. Moschitti, D. Pighin, R. Basili. Tree kernels for semantic role labeling. *Comput. Linguist.*, 34(2), 2008.
- [96] S. Narayanan, C. F. Baker, C. J. Fillmore, M. R. L. Petruck. FrameNet meets the semantic web: Lexical semantics for the web. *ISWC*, 2003.
- [97] F. Naumann, M. Herschel. An Introduction to Duplicate Detection. *Morgan & Claypool*, 2010.
- [98] T. Neumann, G. Weikum. RDF-3X: a RISC-style engine for RDF. *PVLDB*, 1(1), 2008.
- [99] Z. Nie, Y. Ma, S. Shi, J.-R. Wen, W.-Y. Ma. Web object retrieval. *WWW*, 2007.
- [100] P. Palaga, L. Nguyen, U. Leser, J. Hakenberg. High-performance information extraction with Alibaba. *EDBT*, 2009.
- [101] M. Pasca. Towards temporal web search. *SAC*, 2008.
- [102] D. Petkova, W. B. Croft. Hierarchical language models for expert finding in enterprise corpora. *ICTAI*, 2006.
- [103] S. P. Ponzetto, R. Navigli. Large-scale taxonomy mapping for restructuring and integrating Wikipedia. *IJCAI*, 2009.
- [104] S. P. Ponzetto, M. Strube. Deriving a large-scale taxonomy from Wikipedia. *AAAI*, 2007.
- [105] S. P. Ponzetto, M. Strube. WikiTaxonomy: A large scale knowledge resource. *ECAI*, 2008.
- [106] H. Poon, P. Domingos, M. Sumner. A general method for reducing the complexity of relational inference and its application to MCMC. *AAAI*, 2008.
- [107] A. Pugliese, O. Udrea, V. S. Subrahmanian. Scaling RDF with time. *WWW*, 2008.
- [108] F. Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, S. Vaithyanathan. An algebraic approach to rule-based information extraction. *ICDE*, 2008.
- [109] P. Resnik, E. Hardisty. Gibbs sampling for the uninitiated. Technical report, UMIACS, 2009.
- [110] M. Richardson and P. Domingos. Markov Logic Networks. *Machine Learning*, 2006.
- [111] D. Roth, W. Yih. Global Inference for Entity and Relation Identification via a Linear Programming Formulation. *MIT Press*, 2007.
- [112] A. Sahuguet, F. Azavant. Building intelligent web applications using lightweight wrappers. *Data Knowl. Eng.*, 36(3), 2001.
- [113] S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3), 2008.
- [114] S. Sarawagi, W. W. Cohen. Semi-Markov conditional random fields for information extraction. *NIPS*, 2004.
- [115] A. D. Sarma, M. Theobald, J. Widom. LIVE: A lineage-supported versioned DBMS. *SSDBM*, 2010.
- [116] P. Serdyukov, H. Rode, D. Hiemstra. Modeling multi-step relevance propagation for expert finding. *CIKM*, 2008.
- [117] W. Shen, A. Doan, J. F. Naughton, R. Ramakrishnan. Declarative information extraction using Datalog with embedded extraction predicates. *VLDB*, 2007.
- [118] A. Sheth, C. Ramakrishnan. Semantic (web) technology in action: Ontology driven information systems for search, integration and analysis. *IEEE Data Eng. Bull.*, 26, 2003.
- [119] P. Singla, P. Domingos. Entity resolution with Markov Logic. *ICDM*, 2006.
- [120] S. Staab, R. Studer (Eds.). Handbook on Ontologies (2nd Edition). *Springer*, 2009.
- [121] F. M. Suchanek. Automated Construction and Growth of a Large Ontology. *PhD thesis, Saarland University*, 2008.
- [122] F. M. Suchanek, G. Ifrim, G. Weikum. Combining linguistic and statistical analysis to extract relations from web documents. *KDD*, 2006.
- [123] F. M. Suchanek, G. Kasneci, G. Weikum. YAGO: a core of semantic knowledge. *WWW*, 2007.
- [124] F. M. Suchanek, G. Kasneci, G. Weikum. YAGO: A large ontology from Wikipedia and WordNet. *J. Web Sem.*, 6(3), 2008.
- [125] F. M. Suchanek, M. Sozio, G. Weikum. SOFIE: a self-organizing framework for information extraction. *WWW*, 2009.
- [126] C. Sutton, A. McCallum. Introduction to Conditional Random Fields for Relational Learning. *MIT Press*, 2006.
- [127] J. Tappolet, A. Bernstein. Applied temporal RDF: Efficient temporal querying of RDF data with SPARQL. *ESWC*, 2009.
- [128] B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, J. Sperling. NewsStand: a new view on news. *GIS*, 2008.
- [129] M. Theobald, R. Schenkel, G. Weikum. Exploiting structure, annotation, and ontological knowledge for automatic classification of XML data. *WebDB*, 2003.
- [130] G. Tummarello. SIG.MA: Live views on the web of data. *WWW*, 2010.
- [131] O. Udrea, L. Getoor, R. J. Miller. Leveraging data and structure in ontology integration. *SIGMOD*, 2007.
- [132] D. Vallet, H. Zaragoza. Inferring the most important types of a query: a semantic approach. *SIGIR*, 2008.
- [133] M. Verhagen, I. Mani, R. Sauri, R. Knippen, S. B. Jang, J. Littman, A. Rumshisky, J. Phillips, J. Pustejovsky. Automating temporal annotation with TARSQI. *ACL*, 2005.
- [134] R. C. Wang, W. W. Cohen. Language-independent set expansion of named entities using the web. *ICDM*, 2007.
- [135] R. C. Wang, W. W. Cohen. Character-level analysis of semi-structured documents for set expansion. *EMNLP*, 2009.
- [136] Y. Wang, M. Zhu, L. Qu, M. Spaniol, G. Weikum. Timely YAGO: Harvesting, querying, and visualizing temporal knowledge from Wikipedia. *EDBT*, 2010.
- [137] G. Weikum. Harvesting, searching, and ranking knowledge on the web. *WSDM*, 2009.
- [138] D. S. Weld, R. Hoffmann, F. Wu. Using Wikipedia to bootstrap open information extraction. *SIGMOD Record*, 37(4), 2008.
- [139] F. Wu, D. S. Weld. Autonomously semantifying Wikipedia. *CIKM*, 2007.
- [140] F. Wu, D. S. Weld. Automatically refining the Wikipedia infobox ontology. *WWW*, 2008.
- [141] F. Xu, H. Uszkoreit, H. Li. A seed-driven bottom-up machine learning framework for extracting relations of various complexity. *ACL*, 2007.
- [142] A. Yates, M. Banko, M. Broadhead, M. J. Cafarella, O. Etzioni, S. Soderland. TextRunner: Open information extraction on the web. *HLT-NAACL*, 2007.
- [143] Q. Zhang, F. M. Suchanek, L. Yue, G. Weikum. TOB: Timely ontologies for business relations. *WebDB*, 2008.
- [144] J. Zhu, Z. Nie, X. Liu, B. Zhang, J.-R. Wen. StatSnowball: a statistical approach to extracting entity relationships. *WWW*, 2009.