

Open Information Extraction for the Web

Michele Banko

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2009

Program Authorized to Offer Degree: Computer Science and Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Michele Banko

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

Oren Etzioni

Reading Committee:

Oren Etzioni

Alon Halevy

Daniel S. Weld

Date:

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

University of Washington

Abstract

Open Information Extraction for the Web

Michele Banko

Chair of the Supervisory Committee:
Professor Oren Etzioni
Computer Science and Engineering

The World Wide Web contains a significant amount of information expressed using natural language. While unstructured text is often difficult for machines to understand, the field of Information Extraction (IE) offers a way to map textual content into a structured knowledge base. The ability to amass vast quantities of information from Web pages has the potential to increase the power with which a modern search engine can answer complex queries.

IE has traditionally focused on acquiring knowledge about particular relationships within a small collection of domain-specific text. Typically, a target relation is provided to the system as input along with extraction patterns or examples that have been specified by hand. Shifting to a new relation requires a person to create new patterns or examples. This manual labor scales linearly with the number of relations of interest.

The task of extracting information from the Web presents several challenges for existing IE systems. The Web is large and heterogeneous; the number of potentially interesting relations is massive and their identity often unknown. To enable large-scale knowledge acquisition from the Web, this thesis presents *Open Information Extraction*, a novel extraction paradigm that automatically discovers thousands of relations from unstructured text and readily scales to the size and diversity of the Web.

TABLE OF CONTENTS

	Page
List of Figures	iii
List of Tables	v
Chapter 1: Introduction	1
1.1 Thesis Overview	3
1.1.1 Thesis Organization	5
1.2 Advances in IE Automation	5
1.2.1 Weakly-Supervised Systems	6
1.2.2 Self-Supervised Systems	6
Chapter 2: Open Information Extraction	8
2.1 The Feasibility of Relation-Independent Extraction	10
2.2 Designing an Open Extraction System	12
2.2.1 Knowledge Engineering	12
2.2.2 Machine Learning	14
2.2.3 Text Analysis	15
2.3 Related Work	19
Chapter 3: The TextRunner Open Information Extraction System	23
3.1 System Architecture	23
3.1.1 Learner	25
3.1.2 Extractor	27
3.1.3 Assessor	31
3.1.4 Query Processor	33
3.2 Experimental Results	34
3.2.1 Learning an Open Extractor	36
3.2.2 To Parse or Not to Parse	37
3.2.3 Conclusion	39

Chapter 4:	Open Extraction Meets The Web	40
4.1	Experimental Setup	45
4.1.1	Terminology and Parameters	45
4.1.2	Corpora	46
4.1.3	Evaluation Criteria	48
4.2	Experimental Results	49
4.2.1	Relation Extraction	50
4.2.2	The Importance of Redundancy	52
4.2.3	Relation Discovery	55
Chapter 5:	The Relationship Between Open and Traditional IE	64
5.1	Traditional Relation Extraction	65
5.1.1	Experimental Results	66
5.2	Hybrid Relation Extraction	68
5.2.1	Stacked Relation Extraction	69
5.2.2	Experimental Results	70
5.3	The R-TEXTRUNNER System	71
5.3.1	Algorithm	72
5.3.2	Experimental Results	74
Chapter 6:	Conclusions	81
Bibliography	84
Appendix A:	TextRunner Data	93

LIST OF FIGURES

Figure Number	Page
2.1 Natural Language Parsing: The output of a parser when applied to the sentence, “ <i>Jaguar, the luxury auto maker last year sold 1,214 cars in the U.S.</i> ” Nodes of the tree are typically annotated with semantic roles such as subjects (NP-SBJ), objects (NP-OBJ) and phrases indicating time (TMP) and location (LOC).	17
3.1 The TextRunner Open Extraction System: TEXTRUNNER uses a small set of relation-independent heuristics that when applied to parse trees, make it possible to self-supervise the learning of an open extractor. The Extractor outputs a set of relational tuples; synonymous tuples are then found, merged and scored by the Assessor.	24
3.2 Dependency Parsing: Tokens within a sentence are labeled with semantic roles and dependencies from which tuples can be extracted, <i>e.g. (John, hit, ball)</i>	25
3.3 Training TextRunner’s Extractor: TEXTRUNNER uses a domain-independent rule-based algorithm to identify positive and negative examples of English relationships from a parsed corpus of text.	28
3.4 Relation Extraction as Sequence Labeling: A CRF is used to identify the INVENTOROF relationship between <i>Tim Berners-Lee</i> and <i>the WWW</i>	30
3.5 Query Processing: TEXTRUNNER finds answers to the query, <i>What kills bacteria?</i> from multiple documents	35
4.1 A query for “ <i>Napoleon Bonaparte</i> ” illustrates the difference between information found in his Wikipedia summary (top) and facts extracted by TEXTRUNNER (bottom).	42
4.2 Open Extraction from Wikipedia: TEXTRUNNER extracts 32.5 million distinct assertions from 2.5 million Wikipedia articles. 6.1 million of these tuples represent concrete relationships between named entities. The ability to automatically detect synonymous facts about abstract entities remains an open problem.	47

4.3	Open Extraction from the Web: TEXTRUNNER extracts 218 million distinct assertions from 2 or more different sentences in a corpus of over 500 million Web pages (including Wikipedia) . 13.5 million of these tuples represent concrete relationships between named entities. Due to TEXTRUNNER’s redundancy-driven design, the measurements do not include assertions extracted from only one sentence. The ability to detect synonyms of abstract entities and relationships represents an area for future work.	48
4.4	Measuring the Redundancy of Wikipedia and General-Web: Compared to Wikipedia, where only 1.2% of facts are repeated in multiple sentences, the Web contains many different expressions of the same concept. . . .	53
4.5	A fact found once in Wikipedia was found in more than four different sentences on the Web, on average.	54
4.6	The Impact of Redundancy on Precision: While assertions found within at most two different sentences had a precision of 84.3%, precision improved to 87.1% when 20 or more supporting sentences were found, and to 92.3% when at least 200 sentences expressed a single assertion.	55
4.7	The DBpedia Ontology: A hand-built ontology derived from Wikipedia contains 14 relations about objects of type <i>Company</i> . 8 additional relations are inherited from its parent type, <i>Organization</i>	56
5.1	Precision-Recall Tradeoff Between Open and Traditional IE: O-CRF can achieve high precision without any relation-specific input. Recall is lower than R1-CRF, a traditional extractor trained using thousands of examples per relation.	67
5.2	Automatically Improving the Recall of Open IE: Once a set of relation names and reliable instances have been extracted by TEXTRUNNER, R-TEXTRUNNER uses them to improve the recall on a per-relation basis. On average, recall more than doubles when using 1,000 examples per relation and triples when using 10,000 examples.	77
5.3	State-of-the-Art Web IE without the Effort: For a set of 4 relations, R-TEXTRUNNER, a self-supervised Web IE system, uses the knowledge acquired by an Open IE system to automatically improve its performance. Without any manual input, R-TEXTRUNNER identifies a larger number of high-precision assertions than its supervised and weakly-supervised counterparts, which require anywhere from tens to thousands of labeled inputs per relation.	78

LIST OF TABLES

Table Number	Page
2.1 The Contrast Between Traditional and Open IE: Traditional IE methods learn distinct models for individual relations using patterns or labeled instances, thus requiring manual labor that is linear in the number of relations. Open IE learns a single domain-independent extraction model that discovers an unbounded set of relations with only a one-time cost.	10
2.2 Taxonomy of Binary Relationships: Nearly 95% of 500 binary extractions were described using one of eight lexico-syntactic patterns. NP refers to noun phrases, E_i refers to entities, and Prep indicates a preposition.	12
3.1 Open Extraction by Relation Category: O-CRF outperforms O-NB, obtaining nearly double its recall and increased precision. O-CRF’s gains are partly due to its lower false positive rate for relationships categorized as “Other.”	39
4.1 Comparison of TextRunner on Wikipedia and General-Web Corpora: From over 500 million Web pages in the General-Web corpus, TEXTRUNNER extracts instances of ten well-studied relations with an average precision of 92.93%. When applied to the smaller Wikipedia corpus, TEXTRUNNER’s precision is 90.49%, a difference found to be statistically significant with 90% confidence; standard deviation of the mean of differences is 4.0. If tuples containing ambiguous entities are judged as errors, differences in <i>strict precision</i> are not significant. The total number of extractions per-relation is also provided, noting that we do not retain singleton extractions ($f = 1$) from the General-Web corpus.	51
4.2 A Comparison of Web Extraction Systems: The estimated number of entities, relations, facts and classes found by each of TEXTRUNNER, the DBPEDIA ontology, and YAGO.	57
4.3 TextRunner and Yago: YAGO utilizes a set of handcrafted patterns to extract data for a small, fixed set of relations from Wikipedia. TEXTRUNNER learns a relation-independent model to extract data from arbitrary Web pages at a comparable level of precision. The number of facts found by the two systems varies by relation and corpus.	58

4.4	Estimated Number of Relations about Popular Entity Types: <code>TEXTRUNNER</code> , an Open IE system finds an order of magnitude more relations than handcrafted Wikipedia extractors <code>DBPEDIA</code> and <code>YAGO</code>	60
4.5	Relations Associated with Politicians: Using handcrafted rules, <code>DBPEDIA</code> 's ontology contains 32 relations, while <code>YAGO</code> finds 21 relations. <code>TEXTRUNNER</code> finds 630 relations from the general Web in a domain-independent manner. A maximum of 25 results are listed per system.	61
4.6	Relations Associated with Countries: Using handcrafted rules, <code>DBPEDIA</code> 's ontology contains 30 relations, while <code>YAGO</code> finds 32 relations. <code>TEXTRUNNER</code> finds 3632 relations from the general Web in a domain-independent manner. A maximum of 25 results are listed per system.	62
4.7	Relations Associated with Companies: Using handcrafted rules, <code>DBPEDIA</code> 's ontology contains 22 relations, while <code>YAGO</code> finds 17 relations. <code>TEXTRUNNER</code> finds 1478 relations from the general Web in a domain-independent manner. A maximum of 25 results are listed per system.	63
5.1	Summary of IE Corpora: A collection of labeled Web sentences containing instances of four relations. The numbers in parenthesis indicate the number of positive examples.	66
5.2	An Effort to Match Open IE: For 4 relations, a minimum of 4677 hand-tagged examples is needed for <code>R1-CRF</code> to approximately match the precision of <code>O-CRF</code> for each relation. A “*” indicates the use of all available training data; in these cases, <code>R1-CRF</code> was unable to match the precision of <code>O-CRF</code> . . .	68
5.3	Hybrid Information Extraction: A hybrid extractor that uses Open IE improves precision for all relations, at an insignificant cost to recall.	70
5.4	R-TextRunner: Once <code>TEXTRUNNER</code> has discovered a set of relations and trustworthy instances of each, the output can be used to automate additional learning about a particular relation. For the four relations we studied, we provide the phrases most-frequently used by <code>R-TEXTRUNNER</code> to automatically label positive instances for learning. Words have been normalized to their base forms.	79

5.5 **A Comparison of IE Paradigms:** Given a corpus of D documents and R relations to extract, we compare an open extractor, a supervised extractor, a weakly-supervised extractor and a self-supervised extractor trained using the open extractor and the Web. Listed are area under the precision/recall curve (AUC) in the high-precision range (90%-100%). With the exception of the open extractor, each system uses the same underlying extraction algorithm to learn a traditional single-relation extractor. While the labor cost of supervised and weakly-supervised learning is on the order of 10 and 1000 hand-labeled instances per relation, respectively, the amount of manual effort required by the open and self-supervised systems is independent of R . The self-supervised extractor consistently locates a greater number of high-precision extractions than the completely supervised version. 80

ACKNOWLEDGMENTS

I am thankful for the generous support provided by my advisor, Oren Etzoni as well as the members of my doctoral committee: Daniel S. Weld, Alon Halevy and John Gennari. I am grateful to Patrick Allen and Lindsay Michimoto for consistent help and support within the Computer Science Department. Special thanks to Stephen Soderland, Mike Cafarella, Ana-Maria Popescu and Doug Downey, and all past and present members of the KnowItAll group. Finally, thank you Mom, Dad, Frank and Tom for your loving support and patience through all of the hard work.

DEDICATION

In memory of Frank

Chapter 1

INTRODUCTION

While today's Web search engines are useful tools for locating answers to many questions, finding answers to complex queries often requires a substantial effort on the user's part. Consider a student interested in politics who wishes to apply to universities in the eastern United States with award-winning faculty on staff. Relevant information may be scattered over many Web pages – university homepages, news articles announcing recent recipients of the Pulitzer Prize, and an encyclopedia discussing the geography of the U.S. Once a set of Web pages has been found, the user needs to identify the names of several key entities and how they relate. Finally, the student must piece the assorted bits of information together to formulate an answer. For instance, given that Harvard is a university in Boston, Boston is on the eastern coast of the U.S., and that Samantha Power, who teaches at Harvard's Kennedy School of Government, won a Pulitzer Prize, the student may add Harvard to the set of institutions matching the specified criteria. Repeating this process by hand in search of additional candidates is both tedious and error prone.

Traditional search engines cannot automatically provide answers to such complex requests. Instead of aggregating essential information from multiple sources, they return results as a sequential list of documents. Furthermore, even though search engines analyze hyperlinks and anchor text, they remain keyword-based; a page is retrieved only when it (or the corresponding anchor text) matches the words in the user's query. Search engines have eschewed analysis involving entities and relationships because in the past, their identification has not been possible for the size and diversity of information contained on the Web.

Information Extraction (IE), the task of automatically extracting knowledge from text, offers a promising solution for improving Web search by bringing relational structure to unstructured text. At the core of an IE system is an *extractor*, which identifies entities and

relationships expressed using natural language. For example, an extractor might map the sentence “*Hitachi GST was established in 2003 and is headquartered in San Jose*” to the relational tuple $(Hitachi\ GST, Headquarters, San\ Jose)$, which might be represented using RDF or a formal logical language.

Considerable knowledge is necessary to accurately extract these tuples from a broad range of text. To do so, traditional IE systems have thus relied on a significant amount of human involvement. A target relation (*e.g.*, companies and their locations) is provided to the system as input along with hand-crafted extraction patterns or positive and negative instances of the relation. Such inputs are *specific* to the target relation; shifting to a new relation requires a person to manually create new extraction patterns or specify new training examples. This manual labor scales linearly with the number of target relations. Moreover, the user is required to explicitly pre-specify each relation of interest. These systems are clearly not scalable or portable across domains.

Until recently, most work in IE has focused on locating instances of known relationships from small, domain-specific text corpora, such as newswire articles or job postings. Modern IE systems [33, 62, 86] have demonstrated the ability to extract a large number of facts from the Web. However, these systems continue to require relation names as input. Thus, the extraction process has to be run, and rerun, each time a relation of interest is identified.

Despite recent progress in IE, the task of extracting information from the Web presents several challenges for existing systems. The Web is massive and heterogeneous, the relations of interest are unanticipated, and their number can be large. Is it possible to obviate the relation specificity of traditional IE architectures, guaranteeing that extraction of a massive number of relations scales along with the size of the corpus?

This thesis introduces a solution to the above question by investigating the following hypothesis:

We can automatically discover high-quality instances of a large, diverse set of relationships from unstructured Web text using an amount of time and effort that is independent of the number of target relations.

This thesis presents *Open Information Extraction* (Open IE), a new paradigm that moves away from architectures that require relations to be specified prior to query time in favor of a single data-driven process that discovers an unbounded number of relations whose identity need not be known in advance. Open IE makes it possible for users to issue diverse queries over data extracted from the Web.

1.1 Thesis Overview

By addressing the following challenges outlined for extraction from the Web, the development of the Open IE paradigm makes several important contributions to the field of information extraction.

Automation To be able to handle a broad range of queries, a Web IE system must preemptively extract instances of as many relations as possible. While the precise number of relations in the English language is unknown, lexical resources for the English language suggest that the number is in the thousands or higher. VerbNet [46] and PropBank [3], two verb lexicons for the English language, contain 5000 and 3600 semantically distinct entries, respectively. WordNet [56], a broad-coverage catalog of English words, contains over 25,000 verbs and 150,000 nouns.¹

As opposed to traditional IE systems which require manual labor to build a distinct extractor for every relation of interest, our research describes a method for learning one model of how relationships are expressed *in general*. The model is relation-independent, but language-specific. Thus, the cost to develop an Open IE system is incurred once per language, and is independent of the number of target relations in a given language.

Without any relation-specific input, we show that an Open IE system obtains the same precision compared to a traditional extractor trained using hundreds, and sometimes thousands, of labeled examples per relation. While the baseline version of an Open IE system achieves lower recall than traditional extractors, we show that we can leverage high-quality output of the baseline system to automatically learn more about specific relations of interest

¹While not all nouns indicate relationships, *e.g.* “*carrot*” or “*fireplace*,” many nouns such as “*capital*” “*population*” and “*headquarters*” express salient relationships between objects.

as needed, without additional manual labor. With this extension to Open IE, we show that we can increase the number of high-precision extractions found by a factor of 3, and locate an average of 10% more high-quality facts than a state-of-the-art supervised IE system.

Domain-Independence Due to the diversity of genres and topics present in Web text, the ability of an Web IE system to process text in any domain is of critical importance. A handful of approaches to knowledge acquisition [21, 54, 75] have tried to extract relationships by analyzing the full structure and meaning (*i.e.* syntax and semantics) of natural language text. The use of entity recognizers, which attempt to identify instances of a small, fixed set of entity types, such as *Person*, *Location* and *Organization*, is another mechanism commonly employed by traditional IE systems. Although such linguistic analyzers perform well when trained and applied to a particular genre of text, such as financial news articles or biology publications, their ability to handle the heterogeneity of Web text remains uncertain.

We develop an extraction model that learns to capture relational dependencies typically obtained via syntactic and semantic analysis using only domain-independent features that do not require deep linguistic processing. From the use of word-based features to semantic analysis, we explore the use of natural language processing tools in the context of IE, and find that deep linguistic analysis is not necessary for Web-scale extraction. Compared to an extractor that employs full syntactic and semantic processing at extraction time, we demonstrate that an Open IE system can achieve a relative gain in recall of 22% while achieving the same level of precision and running at a speed that is 5 times as fast.

Efficiency A final property necessary for a Web-scale extraction system is the ability to scale to corpora that contain hundreds of billions of documents. Each time a traditional IE system is asked to find instances of a new set of relations, it may be forced to examine a substantial fraction of the documents in the corpus, making system runtime $O(RD)$ for a collection of D documents and R target relations. When D and R are large, as is typically the case on the Web, Open IE’s ability to extract information for all relations at once without having them named explicitly in its input results in a significant scalability advantage over previous IE architectures.

We present `TEXTRUNNER`, a fully implemented Open IE system that supports efficient extraction and exploration of data extracted from a corpus of 500 million Web pages. We report on experiments that measure `TEXTRUNNER`'s ability to find high-quality facts as the size of its corpus size increases by several orders of magnitude. When focusing our study to a set of 10 known relations historically studied by the IE community, we find that `TEXTRUNNER` locates hundreds of thousands of instances with a precision of 92.93% from our corpus of 500 million Web pages.

1.1.1 Thesis Organization

The remainder of Chapter 1 surveys progress in the field leading up to Open IE. Chapter 2 considers the feasibility and design of an Open IE system, and considers existing work related to the open extraction paradigm. Chapter 3 presents `TEXTRUNNER`, the first Open IE system, and studies its ability to extract instances of relations when the number is large and identity unknown. Chapter 4 reports on `TEXTRUNNER`'s performance when applied a Web corpus containing hundreds of millions of documents. Chapter 5 assesses the performance of Open IE relative to previous IE paradigms. We conclude with an overview of possible directions for Open IE.

1.2 Advances in IE Automation

An important step in automating IE was the movement from knowledge-based systems to extractors learned from data. Modern IE, beginning with the works of Soderland [84, 83], Riloff [69] and Moldovan and Kim [48], automatically learned an extractor from a training set in which domain-specific examples were tagged. Keeping pace with progress in machine learning, a diverse set of learning algorithms has been applied to the task of IE, including support vector machines [13, 21, 99], hidden Markov models [36], conditional random fields [63, 23] and Markov logic networks [40].

Nevertheless, the development of suitable training data for supervised IE requires substantial effort and expertise. Systems based on *weakly-supervised learning*, where a human provides a small number of inputs that bootstrap learning over an unlabeled corpus, and *self-supervised learning*, where the system automatically finds and labels its own examples,

further reduced the time required to develop IE systems. We now briefly survey these methods.

1.2.1 *Weakly-Supervised Systems*

Brin [11], Agichtein and Gravano [1], Riloff and Jones [70], Pasca *et al.* [62], and Bunescu and Mooney [12] sought to reduce the amount of manual labor necessary to perform relation-specific extraction. Rather than demand hand-tagged corpora, these *weakly-supervised* IE systems required a user to specify relation-specific knowledge in the form of a small set of seed instances known to satisfy the relation of interest. For instance, by specifying the pairs (*Microsoft, Redmond*), (*Exxon, Irving*) and (*Intel, Santa Clara*) these IE systems learned patterns (e.g. $\langle X \rangle$'s headquarters in $\langle Y \rangle$ and $\langle Y \rangle$ -based $\langle X \rangle$) that identified additional pairs of company names and locations satisfying the HEADQUARTERS(X, Y) relation.

While these systems reduced the amount of required labeled inputs by a significant amount, and can achieve levels of precision and recall on par with fully-supervised systems, the remaining amount of labeling effort becomes non-trivial when the goal is to extract instances of thousands of relations.

1.2.2 *Self-Supervised Systems*

KNOWITALL [33] is a state-of-the-art Web extraction system that addresses the automation challenge by learning to label its own training examples, and tackles issues pertaining to corpus heterogeneity by not relying on deep linguistic analysis or entity recognizers. Given a relation, KNOWITALL used a set of domain-independent patterns to automatically instantiate relation-specific extraction rules. For example, KNOWITALL utilized generic extraction patterns like " $\langle X \rangle$ is a $\langle Y \rangle$ " to find a list of candidate members X of the class Y . When this pattern is used for the class *Country*, for instance, it would match the sentence "*Spain is a southwestern European country located on the Iberian Peninsula,*" and output COUNTRY(*Spain*).

KNOWITALL's extraction patterns were applied to Web pages identified via search-engine queries. The resulting extractions were assigned a probability using information-

theoretic measures derived from search engine hit counts, providing a method of identifying which instantiations were most likely to be bona-fide members of the class. For example, in order to estimate the likelihood that “*China*” is the name of a country, KNOWITALL used automatically generated phrases associated with the class to see if there is a high correlation between the number of documents containing the word “*China*” and those containing the phrase “*countries such as.*” Thus KNOWITALL was able to confidently label China, France, and India as members of the class *Country* while correctly knowing that “*Garth Brooks is a country singer*” does not provide sufficient evidence that “*Garth Brooks*” is the name of a country [30]. Finally, KNOWITALL used a pattern-learning algorithm to acquire relation-specific extraction patterns (*e.g.* “*capital of <country>*”) that led it to extract additional countries. Inspired by KNOWITALL, the URES Web IE system [71], also utilized high-quality output from baseline KNOWITALL to automatically supervise the learning of relation-specific extraction patterns with success.

KNOWITALL and URES are *self-supervised*; instead of utilizing hand-tagged training data, each system selects and labels its own training examples and iteratively bootstraps its learning process. Self-supervised systems are a species of unsupervised systems because they require no hand-tagged training examples. However, unlike classical unsupervised systems, self-supervised systems do utilize labeled examples. Instead of relying on hand-tagged data, self-supervised systems autonomously “roll their own” labeled examples.

KNOWITALL was the first published system to carry out unsupervised, domain-independent, large-scale extraction from Web pages. The first implementation of KNOWITALL required large numbers of search engine queries and Web page downloads; as a result experiments using KNOWITALL often took weeks to complete. This issue was addressed in a subsequent implementation, KNOWITNOW [14]. Despite having made important progress in automating IE at a Web scale, KNOWITALL and KNOWITNOW are relation-specific — the set of relations has to be named by the human user in advance. This is a significant obstacle to open-ended extraction; while processing text one often encounters unanticipated concepts and relations. Furthermore, the extraction process is performed over the entire corpus each time a relation of interest is identified. In the remaining chapters we show how the Open IE paradigm retains KNOWITALL’s benefits but eliminates its inefficiencies.

Chapter 2

OPEN INFORMATION EXTRACTION

This chapter introduces *Open Information Extraction* (Open IE), a novel extraction paradigm that enables domain-independent discovery of relations directly from a large body of text. Open IE has been designed to address the specific challenges outlined for extraction over the Web that were discussed in Chapter 1, namely, *automation*, *domain-independence* and *scalability*.

In order to automate extraction of a large, or possibly unknown, set of relationships, we propose building a single model of how relationships are expressed, in general, in a particular language. The language model, which can be learned automatically or developed by hand using domain-independent methods, serves as the basis of an extractor whose input is a corpus of documents in a single language, and whose output is a set of relational tuples. Compared to traditional relation extraction methods, which separately model every relation of interest in a corpus, Open IE's relation-independent design has several advantages. First, relation-independent extraction eliminates the need for a human system developer to manually identify relations for extraction *a priori*. This ensures that the amount of manual labor required for system development is constant rather than linear in the size of the relation set. Second, the movement away from traditional IE architectures recasts extraction as *relation discovery*, in which the output of the system consists of instances of a potentially unbounded set of relations whose identity need not be known prior to extraction.

Due to the diversity of genres and topics present in the Web corpus, the ability of an Open IE system to process text in any domain is essential. Thus, when considering the set of linguistic cues that are potentially useful for identifying instances of relationships, an open extractor should employ forms of linguistic analysis most likely to exhibit robustness – the ability to analyze any text in the input language. In this chapter, we consider different levels of text analysis that may be useful for open extraction.

A final property of an Open IE system is an ability to scale to massive datasets – corpora such as the Web which may contain hundreds of billions of documents. Traditional IE systems must process a corpus of D documents for every relation of interest. This architecture results in a $O(RD)$ runtime that scales linearly with R , the number of relations. While the exact number of relations in the English language is unknown, we can reasonably estimate the number to be tens of thousands according to existing lexical resources such as WordNet [56], VerbNet [46] and PropBank [3].

Given that R is large, to guarantee scalability along with the size of the document collection, the open extraction process is designed to be independent of the number of relations. Open extraction is simply a function of D , the size of the corpus. Considering that D may be on the order of 1 billion, we believe that an extraction process having a runtime cost of $O(D^2)$ or greater is not feasible. However, an extractor that takes a single, or even k passes over the document collection, where k is fixed and small,¹ would enable extraction of a large number of relations at Web-scale.²

To summarize, the key properties that define an Open IE system are:

- **Relation Discovery:** Automatically discovers a large set of relations from a corpus at once using a process that is constant, rather than linear, in the size of the relation set.
- **Domain Independence:** Developed without the use of domain-specific knowledge, and applicable to arbitrary Web documents in the target language.
- **Scalability:** Scales to Web-size document collections, *e.g.* billions of documents.

Table 2.1 encapsulates the differences between traditional IE and the new Open IE paradigm.

¹That is, k is orders of magnitude smaller than R

²Another cost commonly incurred by an IE system, whether traditional or open, involves post-processing steps such as sorting, duplicate removal and counting of tuples. This requires an additional $O(T \log T)$ cost where T is the number of raw tuples. Since both extraction paradigms are subject to this cost, we omit it from our comparative assessment of traditional and open IE.

	Traditional IE	Open IE
Input	Corpus Labeled Data	Corpus Domain-Independent Knowledge
Relations	Specified in Advance	Discovered Automatically
Development Cost	$O(R)$, R relations	$O(1)$
Runtime	$O(RD)$ D documents, R relations	$O(kD)$ D documents, small, fixed k

Table 2.1: **The Contrast Between Traditional and Open IE:** Traditional IE methods learn distinct models for individual relations using patterns or labeled instances, thus requiring manual labor that is linear in the number of relations. Open IE learns a single domain-independent extraction model that discovers an unbounded set of relations with only a one-time cost.

We have identified the properties necessary for Open IE, yet is our ideal system feasible given the nature of the English language? What would be the best approach for building a scalable, relation-independent extractor for English? Section 2.1 considers the possibility of open extraction based on an empirical study of how humans perceive relationships to be expressed in text. Section 2.2 then weighs the relative strengths and weaknesses of two possible approaches to open extraction: knowledge engineering and machine learning. Section 2.3 concludes the chapter with a survey of existing work that is related to the paradigm of Open IE.

2.1 The Feasibility of Relation-Independent Extraction

Previous work has noted that distinguished relations, such as hypernymy (is-a) and meronymy (part-whole), are often expressed using a small number of lexico-syntactic patterns [41]. These patterns are formed using cues that pertain to generalized word and sentence structures. For example the pattern “NOUNPHRASE₁, including NOUNPHRASE₂” could be used to learn that *Seattle* is an instance of the class *City* from the text, “Some cities, including

Seattle, have increased the number of bike lanes.” The identification of these patterns by hand inspired a body of work in which this initial set of extraction patterns is used to seed a bootstrapping process that automatically acquires additional patterns for is-a or part-whole relations [33, 39, 81].

KNOWITALL demonstrated the ability to extract instances of binary “of” relations from the Web using a small set manually-created relation-independent extraction patterns. A simple pattern such as “NOUNPHRASE₁, *R* of NOUNPHRASE₂” could be used to recognize CEOOF(*Steve Jobs, Apple Computer*) from the text, “*Steve Jobs, the brilliant and forward-thinking CEO of Apple Computer, will give a speech on Tuesday.*” While these patterns cover an important subset of binary relations, they are unable to capture many other ways in which binary relations are expressed, such as through verbs (“*Google acquired YouTube*”) or modifiers (“*Lance Armstrong is the Tour de France winner*”).

It is quite natural to consider whether it is possible to recognize *all* relationships in the general case. How are relationships expressed in English sentences? Might it be possible to build a relation-independent extractor? In this section, we show that many relationships are consistently expressed using a compact set of relation-independent lexico-syntactic patterns, and quantify their frequency based on a sample of 500 sentences from an IE training corpus developed by [12]. Since not all sentences were guaranteed to contain a positive instance of a relationship between entities tagged in the corpus, we sampled sentences at random until the number of sentences containing an explicit relationship reached the desired sample size. For simplicity, we restrict our study to binary relationships.

To characterize how binary relationships are expressed, we carefully studied the labeled relation instances and produced a lexico-syntactic pattern that captured the relation for each instance. Interestingly, we found that 95% of the patterns could be grouped into the categories listed in Table 2.2; this observation leads us to believe that open relation extraction is feasible. While a large proportion of the instances are verb-centric, *i.e.* those patterns expressed by solely a verb (37.8%), the combination of a verb and a preposition (16%), an infinitival phrase (9.4%) and a pair of noun phrases coordinated by the same verb (1%), nearly 36% of the relationships are indicated by nouns and other linguistic phenomena.

Relative Frequency	Category	Simplified Lexico-Syntactic Pattern	Example
37.8	Verb	E_1 Verb E_2	X created Y
22.8	Noun+Prep	E_1 NP Prep E_2	X is birthplace of Y
16.0	Verb+Prep	E_1 Verb Prep E_2	X moved to Y
9.4	Infinitive	E_1 to Verb E_2	X plans to acquire Y
5.2	Modifier	E_1 Verb E_2 Noun	X is Y winner
1.8	Coordinate _n	E_1 (and , - :) E_2 NP	X - Y deal
1.0	Coordinate _v	E_1 (and ,) E_2 Verb	X , Y merge
0.8	Appositive	E_1 NP (: ,)? E_2	X hometown : Y

Table 2.2: **Taxonomy of Binary Relationships:** Nearly 95% of 500 binary extractions were described using one of eight lexico-syntactic patterns. NP refers to noun phrases, E_i refers to entities, and Prep indicates a preposition.

2.2 Designing an Open Extraction System

The goal of an Open IE system is to extract tuples representing all possible relationships among entities in a given text. From the sentence, “Microsoft is headquartered in beautiful Redmond, a city located near Seattle,” an Open IE system should be able to output three distinct tuples — (*Microsoft, is headquartered in, Redmond*), (*Redmond, is-a, city*) and (*Redmond, located near, Seattle*) — using a single extractor. This section considers two classes of approaches to extraction, knowledge engineering and machine learning, and then considers various levels of text analysis that may be useful for extraction under either paradigm.

2.2.1 Knowledge Engineering

While the study of relationships performed in Section 2.1 lend support to the possibility of open extraction, it should be apparent that simply applying the above patterns (or an extended set thereof) is not a sufficient solution in itself due to concerns about both precision

and recall.

The set of manually constructed lexico-syntactic patterns are greatly simplified; they lack the exact conditions under which they will reliably produce a correct extraction. For instance, while many relationships are indicated strictly by a verb, detailed contextual cues are required to determine, exactly which, if any, verb observed in the context of two entities is indicative of a relationship between them. Consider the following two sentences that express the ACQUISITION(X, Y) relationship between the entities *Google* and *YouTube*.

- (1) Google announced Tuesday that it bought YouTube for \$1.65 billion.
- (2) Google has announced the acquisition of YouTube.

An extraction rule applied to the first sentence should recognize that the relationship is indicated by the verb *bought* and not *announced*. In the second sentence, the extractor should find that the relationship is indicated not by the verb *announced*, but rather by the phrase *acquisition of*. We evaluated the simple verb-based pattern “NOUNPHRASE₁ .* Verb .* NOUNPHRASE₂” over the collection of sentences used in our corpus study and found that while recall was relatively high — 76.4% — precision was only 37.8%.

On the other hand, specifying a set of complex conditional rules that obtain a high level of precision without sacrificing recall is a task that is often difficult for humans to perform. Brill and Ngai [10] closely studied the performance of eleven advanced computer science students who were asked to develop rules identifying base noun phrases in text. Given a 25K-word training corpus, the researchers found that while the performance of the top three performing annotators came within 0.5% precision and 1.1% recall of a state-of-the-art supervised rule-learning algorithm trained from the same data, variance was high among the group of annotators. The rules written by the worst-performing writers were up to 5.6% less precise and had as much as 11.9% less recall. Interestingly, Brill and Ngai found that while the best-performing rule-writers were able to propose rules covering high-frequency noun phrases at a level on par with the learning algorithm, the human annotators were significantly outperformed by the machine on low-frequency instances, citing a recall of 63.6% for humans compared to 83.5% for the learning algorithm. The authors attribute this result to either reluctance or inability on behalf of the humans to carefully write rules

that cover only a small number of instances.

2.2.2 Machine Learning

Early approaches to information extraction and other natural language processing tasks were developed using the knowledge of experts. Such systems were designed for a specific domain in a particular language, *e.g.* recognizing details about terrorist incidents reported in text, constructing a dialogue to assist with airline travel reservations, or translating Canadian government transcripts between English and French. The cost of building such systems by hand required a non-trivial amount of effort and expertise, resulting in what is known as the *knowledge engineering bottleneck*.

Due to the difficulties surrounding knowledge-intensive development, data-driven methods have become increasingly popular in recent decades. Machine learning algorithms, designed to enable systems to automatically improve performance on a task using statistical or symbolic methods, have made it possible to reduce the cost of labor involved in developing intelligent applications. Learning algorithms require a set of training examples that describe the task at hand as input, and output a model that can later be applied to new inputs. Although so-called *unsupervised* algorithms aim to learn models without the use of hand-tagged inputs, traditional *supervised* algorithms learn from data that has been provided and annotated by a human.

If we opt for a learning-based approach to Open IE, we will either need to acquire labeled data to be used for supervised learning, or devise an unsupervised method that does not require hand-tagged instances. In the supervised case, to guarantee that we will be able to extract instances of an unbounded set of relations from the Web, it is critical to be able to acquire training data without incurring the $O(R)$ cost usually required by traditional IE systems, with R being the number of relations to extract. Alternatively, one might endeavor to take a *self-supervised* learning approach. Self-supervised systems are a species of unsupervised systems in that they do not require hand-tagged training examples. Unlike classical unsupervised systems, self-supervised systems do utilize labeled examples, yet instead of relying on hand-tagged data, self-supervised systems autonomously “roll their

own” training examples.

Another point of concern relative to the use of learning for Open IE concerns the formulation of the learning problem and the set of features used by the extraction model to describe the task. Relation-independent extraction is notably more difficult than previous learning-based formulations of relation extraction for several reasons. Traditional IE is often framed as a classification problem: Given a sentence S and a relation R , does S assert R between two entities? Such IE systems do not attempt to identify the actual snippets of text that signify the name of a relation, since its identity is already known. In the open extraction task, relation names are not provided in advance. An Open IE system has to locate both the entities believed to participate in a relation, *and* the salient textual cues that provide evidence of the relation between them. Moreover, to ensure high-levels of coverage on a *per-relation* basis, following extraction, the system must identify exactly which relation strings $r_1 \dots r_k$ correspond to a general relation of interest R . We need, for example to deduce that “*'s headquarters in*”, “*is headquartered in*” and “*is based in*” are different ways of expressing HEADQUARTERS(X, Y).

2.2.3 Text Analysis

Information extraction systems can use information resulting from different levels of text analysis, from simple lexical (*e.g.* word-based) features to more complex features derived from grammatical or semantic analysis. Intermediate levels of analysis are provided by *part-of-speech (POS) taggers* and *phrase chunkers*. A POS tagger takes a sequence of words as input, and labels each word with its most likely part of speech, *e.g.* noun, verb, adjective *etc.*. A phrase chunker subsequently uses words and POS tags to divide the sentence into a set of non-overlapping phrases. Data-driven taggers and chunkers for English have been shown to achieve a precision of up to 97% and 94%, respectively [89, 90], and are generally considered to be robust.

Named-entity recognizers (NERs) locate and classify names in text. The set of entity types is typically limited to a small number of classes that are specified in advance (*e.g.* *Person, Location, Organization*). NERs are used by traditional IE systems to manually

bound the type of entities that can legitimately participate in a relationship. For instance, the first argument to HEADQUARTERS(X,Y) must be an *Organization* and the second argument a *Location*. Recent work by Downey [29] demonstrated the ability to recognize complex entities in web text where the set of entity classes is unknown and the names are often difficult to recognize (*e.g.* film or book titles).

A substantial body of work in natural language processing involves the development of *parsers*, which attempt to capture structure and meaning of a single sentence. Given an input sentence, a parser returns a tree or directed graph that captures various levels of linguistic information, including the part-of-speech of each word, the presence of phrases, grammatical structures and semantic roles. The structure and annotations provided by parsers can be useful for identifying relationships between entities within a single sentence. Figure 2.1 depicts the output typically provided by a parser. Examples of relationships expressed by the tree include, but are not limited to, subject-verb-object relationships (*Jaguar, sold, 1,214 cars*), appositives (*Jaguar, is, luxury auto maker*), and location-based relationships (*cars, sold in, the U.S.*).

In 2007, Jiang and Zhai [44], studied a large space of text-based features traditionally used for learned extractors in the single-relation setting. They began with a set of basic unit features that included words, part-of-speech tags and entity types, and then added syntactic categories and semantic dependencies from a parser. After varying the set of features selected for learning and closely studying the impact of each setting on two state-of-the-art algorithms when applied to seven types of relations, Jiang and Zhai concluded that the basic unit features were sufficient to achieve state-of-the-art performance. The use of syntactic and semantic information improved performance only slightly, if at all.

While simple unit features appear to adequately model extraction in the single-relation setting, a relation-independent extraction process makes it difficult to leverage the full set of lexical features typically used when performing extraction one relation at a time. For instance, the presence of the words *company* and *headquarters* will be useful in detecting instances of the HEADQUARTERS(X,Y) relation, but are not useful features for identifying relationships in general. Finally, IE systems typically use entity types as a guide (*e.g.*, the second argument to HEADQUARTERS should be a *Location*). In Open IE, since the relations

```

(S
  (NP-SBJ
    (NP (NNP Jaguar))
    ( , ,)
    (NP (DT the) (NN luxury) (NN auto) (NN maker)))
  (NP-TMP (JJ last) (NN year))
  (VP (VBD sold)
    (NP-OBJ (CD 1,214) (NNS cars))
    (PP-LOC (IN in)
      (NP (DT the) (NNP U.S.)))))

```

Figure 2.1: **Natural Language Parsing:** The output of a parser when applied to the sentence, “*Jaguar, the luxury auto maker last year sold 1,214 cars in the U.S.*” Nodes of the tree are typically annotated with semantic roles such as subjects (NP-SBJ), objects (NP-OBJ) and phrases indicating time (TMP) and location (LOC).

are not known in advance, neither are their argument types.

There are several possible uses for parsers in the context of open extraction. Within a learning-based paradigm, an extractor may use features derived from a parser as input during training, perhaps to compensate for the lack of lexical features. In a knowledge-engineering setting, one could use the output of parsers directly, and develop an algorithm that identifies useful propositions contained in the structure of the parse tree. The extractor could take the form of heuristics or tree-based patterns developed by a linguistic expert in the given language of the corpus, or could be learned by an algorithm. We will report on a few such systems in Section 2.3. Independent of this body of work, we now consider the possible strengths and weaknesses of using parsers for extraction from large, heterogeneous corpora such as the Web.

Parsers fall under one of two paradigms: *grammar-driven* parsers, which are developed by hand using a formal grammar, and *data-driven* parsers, which learn probabilistic models of word-based dependencies from annotated data. One of the greatest challenges for parsers

is robustness, the ability to analyze any input. This is due to the fact that some inputs are not in the language described by the parser’s formal grammar, or adequately represented within the parser’s training data.

Concerns about *domain-adaptability* have long been a point of study and concern for the parsing community. MINIPAR [53], a well-studied broad-coverage English parser, was shown to achieve recall of 81.81% over a set of news articles, but its recall decreased when applied to more complex genres such as fiction (75.29%), memoirs/letters (77.15%) and scholarly prose (79.95%). Ratnaparkhi [66] assessed the cross-domain portability of a maximum-entropy parser and found that when trained on a corpus of Wall Street Journal (WSJ) news articles and tested on sections of the Brown corpus — a dataset comprised of fiction, magazines and journal articles – precision dropped by 6.8% and recall by 6.2% compared to training and testing exclusively on the WSJ. Gildea [38] witnessed similar results when studying a statistical chart parser using the same corpora, citing cross-domain losses 5.8% and 5.6% in precision and recall, respectively.

While the cost of CPUs and memory continue to decrease, thus making it possible to speed-up and distribute computation, we also consider parser *speed* as a factor in our design space. The grammar-based MINIPAR parser reportedly processes 500 words of newspaper text per second on a 700MHz Pentium III with 500MB memory; its output was found to be 89% accurate on this corpus. By comparison, while a statistical parser developed this year by Sagae and Tsujii [74] achieved the same accuracy with better recall (88.5% compared to 81.8%), its data-driven algorithm required nearly double the CPU time.

One of the most widely-used and accurate statistical parsers, which was developed by Klein and Manning [50], parses newswire text with an accuracy of 91.0%. However, this gain in accuracy comes at a cost to speed. On a 750MHz Pentium III with 2GB of RAM, their parser requires an average of 90 seconds for sentences up to 40 words long, which is a full three orders of magnitude slower than MINIPAR. Even on short sentences of 20 words, Klein and Manning’s parser takes an average of 14 seconds.

As of December 2008, the English version of Wikipedia alone contains approximately 2.5 million articles consisting of a total of 1 billion words [93]. Based on previously reported parser measurements, it would require roughly 23 CPUs to parse all of the sentences in the

English Wikipedia corpus in a single day with MINIPAR, and over 8000 CPUs using Klein and Manning’s parser.³ To parse a corpus of 1 billion articles, one would need nearly ten thousand machines to be able to process the collection on a daily basis using the faster of the two parsers, or a few thousand machines to process it on a weekly basis. The resources required to parse massive bodies of text is impractical for all but a few organizations.

2.3 *Related Work*

A handful of recent efforts have sought to undertake the extraction of many relations simultaneously, indicating a growing interest in the problem. We now describe several bodies of work that are related to the paradigm of Open IE, noting similarities and differences as appropriate.

KNEXT [75] was developed to glean “general world knowledge” from English text. As with Open IE, KNEXT aims to acquire a broad range of knowledge at once rather than focus on a predetermined domain. Yet unlike IE, where the focus is on finding instances of relationships among real-world entities, the goal of KNEXT is to obtain what can be described as “common sense.” Whereas IE systems output relational data in the form of tuples, database records or instantiated templates, KNEXT composes a body of abstract formulas describing the world using formal logic. For example, given the sentence, “*Mary entered Jane’s room while she slept, bringing back her washed clothes.*” KNEXT proposes quantified logical propositions that express the following statements:

A named-entity may enter a room.
 A female-individual may have a room.
 A female-individual may sleep.
 A female-individual may have clothes.
 Clothes can be washed.

KNEXT operates by applying a set of general rules to the output of a parser. The set of 80 rules were engineered by hand to transform parse-tree fragments into standalone logical

³One can assume some speed-ups are possible as the performance of CPUs has improved in the last few years.

propositions. The ordered set of rules was designed to recognize instances of phrase-structure and semantic patterns, combine and abstract the patterns into a logical representation, and filter out malformed or vacuous propositions. This rule-based algorithm was observed to yield an average of 2.47 propositions per sentence. Following post-processing and removal of duplicate propositions, KNEXT’s extraction rate was found to output roughly 1.78 unique propositions per sentence. Unlike most systems that acquire knowledge from data, KNEXT does not use corpus statistics to assess the likelihood that a proposition is correct. Filtering out any remaining ill-formed propositions is left strictly to human judgment.

An initial version of KNEXT was tailored to the Penn Treebank, a small corpus consisting of 1.6 words of hand-parsed material, making it possible to circumvent any challenges associated with parsing accuracy. Human assessors found about 60% of statements found by KNEXT from the gold-standard parses of the Penn Treebank could be considered a “reasonable general claim” by any judge; fewer propositions were unanimously judged to be reasonable. In 2008, KNEXT was extended to analyze arbitrary inputs using a parser trained from the Penn Treebank [32], and subsequently deployed over a Web corpus of nearly 11.7 million pages. On average, the percentage of propositions judged to be reasonable from this corpus was between 50% and 60%.

There are at least two additional systems belonging to the species of extractors which employ a parser along with a set of hand-crafted rules to harvest a large number of relationships from text. Inspired by the vision of KNEXT, Clark *et al.* [18], applied a parser to a textual corpus of 800,000 news articles and yielded 1.1 million extractions. Instead of manipulating parser fragments into a complex logical representation, Clark’s system outputs knowledge as relational tuples. The set of relation types was limited to subject-verb-object relationships.

The DIRT system [54] used the MINIPAR parser to produce tuples relating two entities in a single sentence. The overall goal of DIRT was not IE per se, but to discover rules from text that could be used to infer whether the meaning of two relational statements is similar (*e.g.* $X \text{ wrote } Y \approx X \text{ is the author of } Y$ and $X \text{ manufactures } Y \approx X\text{'s } Y \text{ factory}$). The system was applied to a 1GB collection of newspaper text from which it extracted 231,000 unique tuples. While the accuracy of the underlying tuple extraction algorithm was not explicitly

measured, the authors found that that accuracy of verb-based tuples was generally much higher than extractions rooted in noun-based relationships. DIRT’s overall ability to learn inference rules varied highly depending on the relation under consideration.

Independent of our work, Shinyama and Sekine [78] developed an unsupervised extraction process described as *unrestricted relation discovery*. Given a collection of documents, the system first clusters the articles using a bag-of-words document representation. Ideally, this step partitions the corpus into sets of articles believed to contain entities bearing similar relationships. Within each cluster, the system then performs named-entity recognition, reference resolution and linguistic parsing, and uses the output to form relational patterns used as features for an additional “meta-clustering” stage. Meta-clustering is computed in pairwise fashion over the set of entities found in the document cluster under consideration. Its output is a set of instances believed to participate in the same relationship, *e.g.* the relationship among a person, company and job-title involved in a hiring event.

While the work of Shinyama and Sekine pursues the important goal of avoiding relation-specificity, it is unlikely to meet the scalability requirement necessary to process the Web. Their system, which uses pairwise vector-space clustering, initially requires an $O(D^2)$ effort where D is the number of documents. Each document assigned to a cluster is then subject to linguistic processing, resulting in another pass through the set of input documents. Finally, each cluster is subject to an additional clustering process on the order of $O(E^2)$ where E is the number of entities. By focusing on newswire articles and clustering only documents authored on the same day, the authors have been able to reduce the cost of clustering somewhat. From a collection of 28,000 newspaper articles, Shinyama and Sekine were able to discover 101 relations, of which roughly 65% were judged to be correct. For a corpus containing tens of thousands of documents, the relation discovery process was measured to take an average of 10 hours using a single 2.4GHz CPU with 4GB of memory.

A final class of systems considered here involves the extraction of structured data exclusively from Wikipedia. The Intelligence in Wikipedia (IWP) project [92] was designed to exploit the fact that each article in Wikipedia corresponds to a primary object and that many articles contain infoboxes, tabular summaries of the most important attributes (and their values) for these objects. IWP engages in a self-supervised learning process in which it

obtains a large set of relations to extract from the infoboxes. Using the value of infobox attributes to match sentences in the article, IWP trains an extractor for each attribute. IWP’s extractors have been demonstrated to achieve high levels of precision and recall for popular attributes. For instance, it can recognize attributes of in the class *U.S. County* with 97.3% precision and 95.9% recall. Unfortunately, some infoboxes may be partially instantiated, and many articles lack infoboxes altogether. By autonomously learning a taxonomy over infobox classes, and constructing schema mappings between the attributes of parent/child classes, IWP uses shrinkage to improve both recall and precision. Once extractors have been successfully learned, IWP can extract values from general Web pages in order to supplement results which may not be present in Wikipedia.

YAGO [86] is another system that uses properties specific to Wikipedia in order to extract a large set of relational data. YAGO performs extraction using a combination of manually developed rules and heuristic methods. In addition to infoboxes, it also uses list-like information contained within Wikipedia category pages (*e.g.* a page containing the names of all winners of the Nobel Prize). While YAGO will be considered in more detail in Chapter 4, it is important to note here that while IWP can learn to extract values for *any* infobox attribute, YAGO’s heuristic-based approach limits the number of relations that can be extracted to a predefined set.

While IWP and YAGO have each amassed an impressive collection of knowledge from Wikipedia, the set of relations they are capable of extracting is limited to the relations contained in infoboxes and lists (and in the case of YAGO, only a pre-specified subset thereof). In 2008, the developers of IWP estimated that tabular summaries about *Companies* contain 83 attributes, some of which are duplicates (*e.g.* *type* and *company_type*). Although infoboxes can provide key definitional attributes of a company such as its location, the date on which it was founded and the name of its current CEO, the summaries fail to capture many other interesting relations present in unstructured natural language text — news about an impending acquisition, plans to offer a new product, or popular opinions of the company.

Chapter 3

THE TEXTRUNNER OPEN INFORMATION EXTRACTION SYSTEM

This chapter presents `TEXTRUNNER`, the first open information extraction system. `TEXTRUNNER` is a fully-implemented system that extracts relational tuples from hundreds of millions of Web pages. Section 3.1 begins with a detailed description of `TEXTRUNNER`'s architecture and considers how each component addresses each of the challenges outlined in Chapter 1. Section 3.2 reports on experiments that measure `TEXTRUNNER`'s ability to extract instances of relationships when their number is large and identity unknown.

3.1 System Architecture

`TEXTRUNNER` consists of four key modules:

- **Learner:** Given a small corpus and set of relation-independent heuristics, the Learner outputs a single extraction model of English relationships. While the model is language-specific, it is relation-independent.
- **Extractor:** The Extractor makes a single pass over the entire corpus to extract tuples for all possible relations.
- **Assessor:** The Assessor identifies instances describing the same real-world object or relation using different names using an unsupervised algorithm [98], and assigns each tuple a score based on a redundancy-based method.
- **Query Processor:** The Query Processor indexes `TEXTRUNNER`'s output using a distributed index, supporting efficient exploration of extracted facts via user queries.

`TEXTRUNNER`'s extraction pipeline is depicted in Figure 3.1. We now describe each module in more detail.

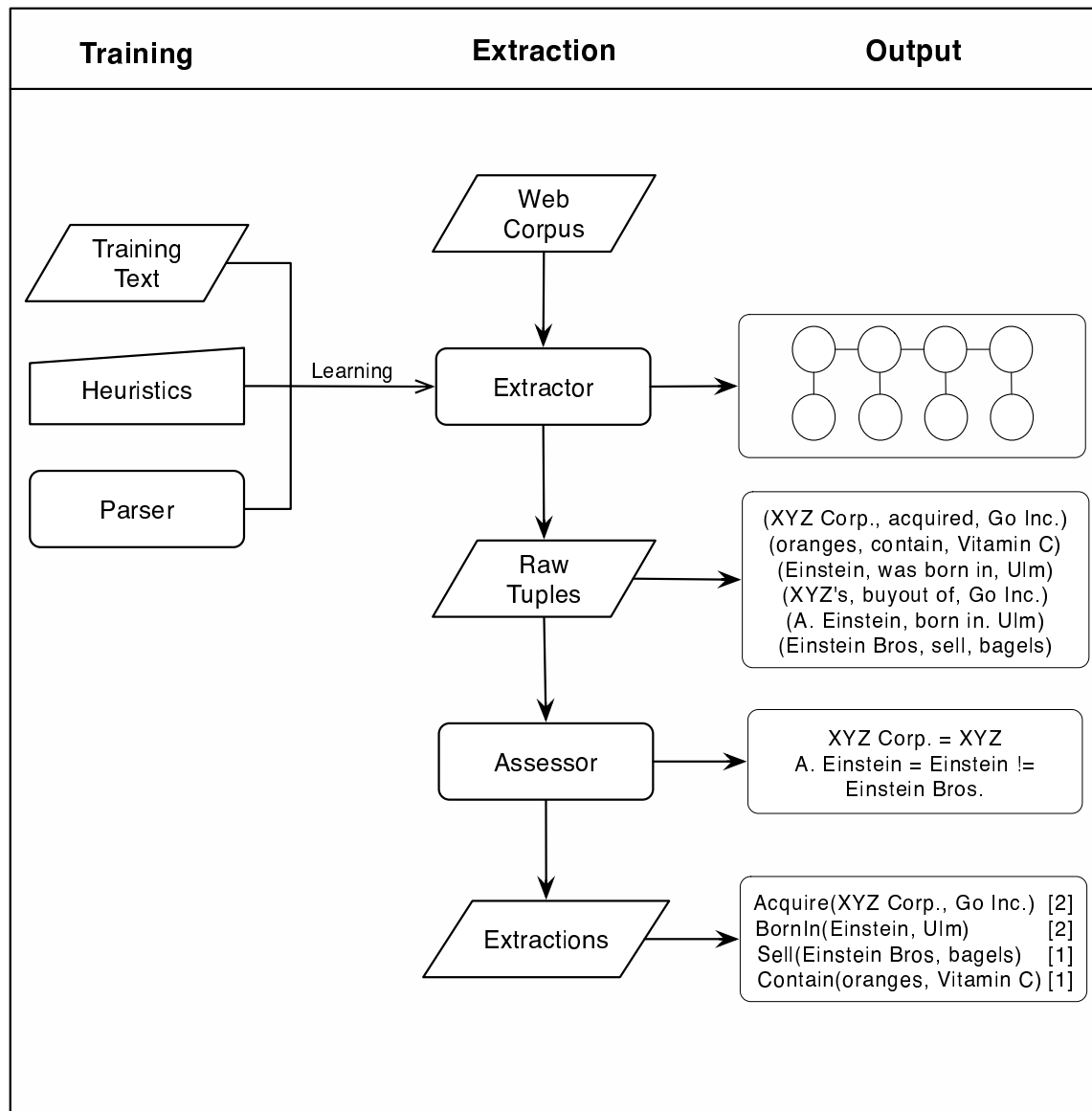


Figure 3.1: **The TextRunner Open Extraction System:** TEXTRUNNER uses a small set of relation-independent heuristics that when applied to parse trees, make it possible to self-supervise the learning of an open extractor. The Extractor outputs a set of relational tuples; synonymous tuples are then found, merged and scored by the Assessor.

3.1.1 Learner

TEXTRUNNER’s self-supervised Learner operates in two steps. First, it automatically identifies and labels its own training examples as positive or negative instances of possible relationships between entities. Second, it uses this labeled data to train the Extractor.

TEXTRUNNER is trained from a set of relational tuples where the relation can be *any* relationship between entities. How might we acquire a set of training examples that is relation-independent without undergoing a large effort to label them by hand?

Natural language parsers, which were discussed in Section 2.2.3, are widely used to determine grammatical structure and semantic roles present in sentences. *Syntactic parsers* provide an analysis of a sentence’s grammatical structure, returning a tree-based hierarchy of all words in the input. An example was given previously in Figure 2.1. *Dependency parsers* locate instances of semantic relationships between words, forming a directed graph that connects all words in the input. As shown in Figure 3.2, examples of relationships found by dependency parsers, include the subject relation ($John \leftarrow hit$), the object relation ($hit \rightarrow ball$) and phrasal modification ($hit \rightarrow with \rightarrow bat$). Some parsers return both syntax and dependency information at once.

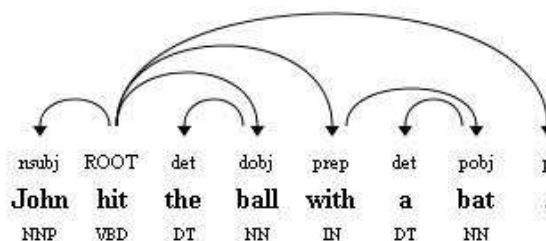


Figure 3.2: **Dependency Parsing:** Tokens within a sentence are labeled with semantic roles and dependencies from which tuples can be extracted, *e.g.* ($John, hit, ball$)

We hypothesize that the use of parsers may not be useful for direct extraction over the Web for several reasons. The first reason concerns difficulties pertaining to domain adaptability and the heterogeneity of the Web corpus, as reported in Section 2.2.3. The second can be attributed to the recent success of *redundancy-based methods* within the field

of natural language processing [9]. Rather than use “deep” linguistic analyzers over small, specific corpora, these systems use only “shallow” features that can be derived from words and part-of-speech tags and instead rely on the redundancy of the input corpus – the existence of multiple, differently phrased statements of the same statement. As the size of the input corpus increases, it becomes more likely that easy-to-understand forms of a statement are present. For example, consider the transparency of the knowledge expressed by “*President Lincoln was killed by John Wilkes Booth*” compared to a corpus which contains only the text, “*Booth attacked Lincoln at Ford’s Theater. He died the next morning from a single gunshot wound to the back of the head, becoming the first American president to be assassinated.*”

Given that our goal is to apply our Open IE system to the Web, a corpus that is both redundant and large, we designed TEXTRUNNER to use features that are fast and easy to compute at extraction time. Instead of applying a parser directly to the Web corpus, we are optimistic that we can use a set of trusted parse trees to *train* an extractor using a one-time self-supervised procedure. The key to building this extractor is that we model the positive and negative relation instances identified by the parser using only features that do not depend on syntactic or semantic analysis at extraction time. Thus, the learned extractor can be thought of as a system that approximates the behavior of the parser-based algorithm using lower-level signals.

Prior to full-scale relation extraction, TEXTRUNNER applies a handful of relation-independent heuristics to a set of parse trees and obtains a set of labeled examples in the form of relational tuples. The heuristics were designed by hand to capture relational dependencies obtained via syntactic parsing and semantic role labeling. The full algorithm is given in Figure 3.3. Examples of heuristics used to identify positive examples include the extraction of noun phrases participating in a subject-verb-object relationship (lines 6-7) and predicate-argument structures expressing location, time, manner, direction, *etc.* (lines 10-11).¹ For instance, “*Einstein received the Nobel Prize in 1921*” yields two positive tuples: (*Einstein, receive, Nobel Prize*) and (*Einstein, receive (arg2) in, 1921*). An example of a heuristic

¹Semantic roles are identified according to the guidelines of the Penn Treebank as described in <ftp://ftp.cis.upenn.edu/pub/treebank/doc/arpa94.ps.gz>

that locates negative examples is objects that cross the boundary of a clause (line 8), *e.g.* “*He studied Einstein’s work when visiting Germany*” yields a negative instance: (*Einstein’s work, visiting, Germany*). Another useful rule for identifying negative examples is detecting objects that are separated by many words (line 1), or connected by a long chain of dependencies, which typically do not capture salient relationships. The algorithm is designed to target relationships that can be found with high precision; therefore TEXTRUNNER does not extract modification-based relationships such as those expressed in the phrase “*Honda with sunroof*” or “*city of 1 million citizens*” (line 3), or those expressed with punctuation (*e.g.* “*New York City, New York*”).

As the input to TEXTRUNNER’s training process, we used the Penn Treebank, a set of manually constructed parse trees for approximately 42,000 sentences. The Penn Treebank is annotated with both syntactic and semantic information. Applying the domain-independent rule-based algorithm to this corpus yielded roughly 180,000 examples, 15% of which were automatically labeled as positive instances.

The key to TEXTRUNNER’s ability to train a domain-independent extraction model can be attributed to two characteristics of the training process. The features ultimately used to describe the set of labeled examples can be extracted without subsequent syntactic or semantic analysis. Furthermore, the model output by the Learner contains no relation-specific or lexical features. Feature extraction is described in detail in Section 3.1.2.

3.1.2 *Extractor*

TEXTRUNNER learns to identify spans of tokens believed to indicate explicit mentions of relationships between entities. For each sentence in the input corpus, TEXTRUNNER performs entity identification using a maximum-entropy-based model of phrases. The Learner is then used to identify possible instances of relations between each possible entity pair. We developed two different open extractors; the first treated Open IE as a classification problem, and the other as a sequence labeling task. The remainder of this section describes both extraction models and the set of features used by both.

Given a sentence S and its parse tree T , let

$e1$:= a noun phrase in S
 $e2$:= a noun phrase in S such that $e1$ occurs before $e2$ in S
 R := the set of tokens between $e1$ and $e2$
 C := the label of the proposed example

```

1  Return null if NumTokens( $R$ ) >  $M$ , exclusive of tokens
   belonging to other noun phrases in  $R$ 

   // Require verb in the context of extracted entities.
2   $C$  := negative if (not ContainsVerb( $R$ ))

   // If  $e1$  is parent of  $e2$  label as negative
3   $C$  := negative if IsParentOf( $e1$ ,  $e2$ ,  $T$ )

   // Find lowest ancestor of  $e1$  and  $e2$  in  $T$ 
4   $A$  := LowestCommonAncestor( $e1$ ,  $e2$ ,  $T$ )

5  If  $A$  is labeled a sentence or clause node {

   //  $e1$  must be the subject of the sentence/clause
6   $C$  := negative if (not IsSubjectOf( $e1$ ,  $A$ ))

   //  $e1$  must be the head of the sentence/clause
7   $C$  := negative if (not IsHeadOf( $e1$ ,  $A$ ))

   // Limit relationships that span sentence-like
   boundaries (e.g.  $S$ ,  $S$ -BAR) in the tree  $T$ 
8   $C$  := negative if (CrossSentenceBoundary( $e1$ ,  $e2$ ,  $T$ ))

   // Restrict preposition-based relationships
   // to a set of semantic roles: (time, location,
9  // manner, direction, extent, adjunct)
   If  $e2$  is the object of a prepositional phrase,  $PP$  {
10      $C$  := positive if (IsValidSemanticRole( $e2$ ))
11     Else  $C$  := negative
   }

12   Else {
13      $C$  := positive
14      $R$  := normalize( $R$ ) // Find head of  $A$ , the base form
                        // of the relationship
   }
   }

15 Else  $C$  := negative

16 Return ( $C$ ,  $e1$ ,  $R$ ,  $e2$ )

```

Figure 3.3: **Training TextRunner’s Extractor:** TEXTRUNNER uses a domain-independent rule-based algorithm to identify positive and negative examples of English relationships from a parsed corpus of text.

O-NB: Open Extraction as Classification

We first treated open extraction as a classification problem [5], using the set of self-labeled examples to learn a Naive Bayes classifier that predicted whether heuristically-chosen tokens surrounding two entities indicated a relationship or not. Given a pair of entities within a sentence, candidate relationships were found by examining tokens in the immediate context and eliminating non-essential items using evidence from the phrase chunker. Descriptive modifiers such as adverbs and adjectives are dropped (*e.g.* “*definitely developed*” is reduced to “*developed*”). Each candidate tuple is presented to the classifier and retained if it is hypothesized to be a positive relation instance with sufficiently high probability. For the remainder of this paper, we refer to this Open IE model as O-NB.

O-CRF: Open Extraction as Sequence Labeling

Whereas classifiers predict the label of a single variable, graphical models model multiple, interdependent variables. Conditional Random Fields (CRFs) [52], are undirected graphical models trained to maximize the conditional probability of a finite set of labels Y given a set of input observations X . By making a first-order Markov assumption about the dependencies among the output variables Y , and arranging variables sequentially in a linear chain, information extraction can be treated as a sequence labeling problem – the task of assigning a single label to each element in a sequence. Linear-chain CRFs have been applied to a variety of sequential text processing tasks including named-entity recognition, part-of-speech tagging, word segmentation, semantic role identification, and traditional forms of relation extraction [23]. CRFs have been shown to outperform other popular species of graphical models such as Hidden Markov Models (HMMs) and Maximum Entropy Markov Models (MEMMs) because instead of training to predict each label independently, CRFs are trained to get the entire sequence correct.

In light of the success of CRFs, we developed an extractor, referred to as O-CRF [7], which uses a second-order linear chain CRF to learn whether sequences of tokens are part of a salient relation or not. Following entity identification, each pair of entities appearing no more than a maximum number of words apart and their surrounding context are considered

as possible evidence of a relation. The entity pair serves to anchor each end of a linear-chain CRF, and both entities in the pair are assigned a fixed label of ENT. Tokens in the surrounding context are labeled using the BIO encoding widely-used for natural language tasks [65], where each token is labeled as B-X, I-X or O. B-X means “begin a phrase of type X,” I-X means “continue a phrase of type X” and O means “not in a phrase.” We use the formalism to labeling textual cues believed to explicitly indicate a relation, as illustrated in Figure 3.4.

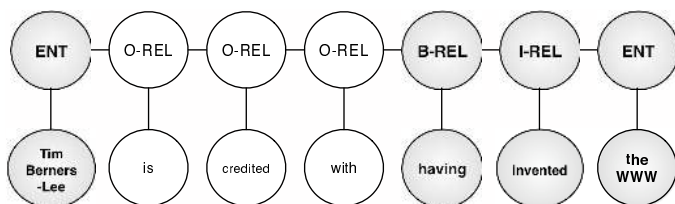


Figure 3.4: **Relation Extraction as Sequence Labeling:** A CRF is used to identify the INVENTOROF relationship between *Tim Berners-Lee* and *the WWW*.

Another advantage of using CRFs is that the formulation makes it possible to model relations possessing more than two arguments in an elegant manner. For example, the following labeling extends the ACQUISITION(X, Y) relation to include the price of the sale, yielding a tuple with three arguments: (*Google*, *acquire (arg2) for*, *YouTube*, *\$1.65 billion*):

```
Google/ENT announced/O that/O it/O acquired/B-REL
YouTube/B-NP for/B-REL an/ENT astonishing/ENT $1.65_billion/ENT
```

O-CRF was built using the CRF implementation provided by MALLET [55], as well as part-of-speech tagging and phrase-chunking tools available from OPENNLP.²

Features

The set of features used by O-NB and O-CRF is similar to those used by other state-of-the-art relation extraction systems [12, 63] with a few exceptions. Within the context of the

²<http://opennlp.sourceforge.net>

entities under consideration for extraction, TEXTRUNNER models features include words, part-of-speech tags and phrase chunks, regular expressions (*e.g.* detecting capitalization, punctuation, *etc.*), and conjunctions of features occurring in adjacent positions within a fixed window.³ A unique aspect of TEXTRUNNER’s extraction model is that it is *unlexicalized*. That is, it models words belonging only to *closed classes* (*e.g.* prepositions and determiners) but not function words such as verbs or nouns. Also, unlike most IE systems, O-NB and O-CRF do not try to recognize semantic classes of entities.

Relation-independent extraction forces a few limitations, most of which are shared with other systems that perform extraction from natural language text. First, O-NB and O-CRF only extract relations that are explicitly mentioned in the text; implicit relationships that could be inferred from the text would need to be inferred from open extractions. Second, O-NB and O-CRF focus on relationships that are primarily word-based, and not indicated solely from punctuation (*e.g.* *Seattle, WA*) or document structure (*e.g.* lists and tables). Finally, relations must occur between entity names within the same sentence.

3.1.3 Assessor

Once the extractor has been applied to the input corpus, TEXTRUNNER prepares its output for indexing using a few post-processing steps. This procedure normalizes the raw tuples, identifies extractions believed to express the same fact, and assigns them a score using a redundancy-based method. We now describe each step in more detail.

Normalization

TEXTRUNNER unifies extracted entity and relation names using a simple normalization routine. All tokens are stemmed into their base forms using a morphological analyzer [37]. Given a hypothesized part of speech tag, the stemmer identifies, for instance, that “*invent*” is the root of the words “*invented*” and “*invents*” and that “*companies*” maps to “*company*.”

TEXTRUNNER further simplifies the names of entities by omitting tokens from noun phrases that potentially lead to overspecification. Entities consisting of common noun

³TEXTRUNNER uses a window of size 6, due to the fact that it does not attempt to model long-distance relationships.

phrases are reduced to their lexical heads using a set of head-finding rules developed and used widely by the parsing community [19]. For example, “*Young scientists from many universities are studying exciting new technologies*” is analyzed as “*Scientists are studying technologies*”). While these heuristics are sufficient in many cases, important information can sometimes be lost, such as that pertaining to quantification (e.g. “*most people*” is reduced to “*people*”) and certain modifiers (e.g. “*green technology* becomes “*technology*”). We hope to address this issue in future work.

Synonym Resolution

IE systems often locate assertions that refer to the same real-world object or relation using different names. For example, the tuples (*Washington D.C., is capital of, U.S.*) and (*D.C., is capital city of, America*) refer to the same fact. Identifying synonymous tuples is critical for high-quality information extraction. End-users of an IE system may face low recall if the system fails to recognize all expressions of a relation in response to a query about a given relation. Another risk is that users are presented with redundant information instead of a concise display.

A study by Yates [97] empirically quantified the importance of *synonym resolution*, the task of identifying synonymous entity and relation names. He examined a large set of TEXTRUNNER’s extractions and found that the top 80 most frequently extracted objects were described by an average of 2.9 different names, with some having as many as 10 names. The 100 most frequent relations had an average of 4.9 synonyms.

Yates developed the RESOLVER algorithm [98], to meet the challenges of information extraction over the Web. RESOLVER is an unsupervised, domain-independent algorithm that runs in time $O(kN \log N)$ where N is the number of extractions and k is the maximum number of synonyms per word.⁴ It uses a probabilistic model to predict the likelihood that two strings refer to the same item based on string-similarity and shared relational attributes. Experiments have shown that RESOLVER outperforms previous approaches to synonym resolution, finding entity synonyms with 78% precision and 68% recall, and relation

⁴Based on empirical analysis, RESOLVER finds that $k = 10$ is sufficient.

synonyms with 90% precision and 35% recall.

Once TEXTRUNNER’s set of raw tuples has been normalized, sorted and duplicates removed, RESOLVER is applied to the set of collated tuples. The resulting output serves as the final set of extractions that is assigned a score and then indexed by TEXTRUNNER’s query processor.

Assessment

Following normalization and synonym resolution, TEXTRUNNER automatically merges tuples in which the entities and relations are identical. The system counts the number of distinct sentences from which each extraction was found. These counts serve as a measure of confidence that a tuple is a correct instance of a relation among entities. Extractions with a count of one are not added to the system’s knowledge base.

The task of sorting tuples and identifying duplicate sentences can become memory-intensive as the number of extracted tuples increases. However, use of the MapReduce framework [27], a programming model that facilitates large-scale distributed processing, has made it possible to efficiently process output from large corpora.

3.1.4 Query Processor

TEXTRUNNER is capable of responding to queries over millions of tuples at interactive speeds due to an inverted index⁵ distributed over a pool of machines. This index is analogous to a standard inverted index computed over corpus documents for document retrieval. In a standard inverted index, each corpus term points to a list of all the documents in which that term appears. In TEXTRUNNER’s inverted index, each entity found in an extracted triple points to a list of all the triples in which it appears, and similarly for relations.

The efficient indexing of tuple in TEXTRUNNER means that when a user (or application) wants to access a subset of tuples by naming one or more of its elements, the relevant subset can be retrieved in a manner of seconds, and irrelevant extractions remain unrevealed to the user.

⁵The inverted index is built using Lucene, an open source search engine.

TEXTRUNNER’s tuple index enables *relational Web search* [15] – search through a large entity-relationship graph that is automatically derived from the text of Web pages. Each node in the graph corresponds to an entity found by TEXTRUNNER, with edges representing relationships between entities.

While traditional search engines analyze hyperlinks and anchor text, they remain keyword-based and fail to aggregate information contained over multiple documents. TEXTRUNNER makes it possible for a user to issue complex relational queries that are not currently possible using today’s search engines, including:

- **Relationship Queries:** Find the relationship(s) between two objects (*e.g.* the relationship between *Bill Clinton* and *Justice Ginsberg*).
- **Factoid Queries:** Find the answer(s) to a query having a short, factual answer (*e.g.* *What kills bacteria?*, which is depicted in Figure 3.5).
- **Qualified List Queries:** Retrieve a list of objects that share multiple properties (*e.g.* *British spy novelists*). An algorithm for computing answers to qualified list queries using TEXTRUNNER was presented in [15].
- **Unnamed-Item Queries:** Qualified-List queries that aim to locate a single object whose name the user cannot recall (*e.g.* *the 40th president of the U.S.*).


3.2 Experimental Results

In Section 2.1 we found that that English relationships (in the binary case) can be consistently described by a set of linguistic patterns. The precise conditions under which they apply, however, are difficult to specify by hand without making significant tradeoffs between precision and recall. Is it, then, possible to *learn* a model of English relationships? Is deep linguistic parsing necessary for open extraction? This section measures the performance of several relation-independent extraction algorithms, when the number of relationships is large and unknown prior to extraction. We assess the behavior of two learned open extractors as well as a hand-built extractor designed to identify instances of binary relationships.

TextRunner Search Results - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Back Forward Reload Stop Home <http://turingc.cs.washington.edu:7125/TextRun> Go Search

 **TextRunner Search**

Retrieved **2849** results for **kills** in the predicate and **bacteria** in argument 2.
 Grouping results by predicate. Group by: [argument 1](#) | [argument 2](#)

kills - 31 results

the new antibiotics (69), Benzoyl peroxide (47), Chlorine (36), [119 more...](#) **kills bacteria**

Heat (27) **kills** the beneficial **bacteria**
 Amoxicillin (26) **kills bacteria**
 ozone (24) **kills** any **bacteria**
 Penicillin (23) **kills** the pneumococcal **bacteria**
 Oxygen (16) **kills** anaerobic **bacteria**
 Honey (12) **kills** the **bacteria**
 Cooking (12) **kills** these **bacteria**
 The process (11) **kills** other **bacteria**
 Irradiation (11) **kills** most harmful **bacteria**
 Garlic (9) **kills** the bad **bacteria**
 Alcohol (9) **kills** the **bacteria**

Search again:

Argument 1

Predicate

Argument 2

examples of "bacteria":

e. coli (13)
 salmonella (12)

Jump to:

[kills \(31\)](#)
[will kill \(7\)](#)

Read turingc.cs.washington.edu

Figure 3.5: **Query Processing:** TEXTRUNNER finds answers to the query, *What kills bacteria?* from multiple documents

- We measure the precision and recall O-NB and O-CRF, two distinct self-supervised learners developed as part of the TEXTRUNNER open extraction system. We show that without any relation-specific input, O-CRF extracts instances of binary relationships with high precision and a recall that nearly doubles that of O-NB.
- We quantify the performance of O-CRF, TEXTRUNNER’s best performing extractor, relative to O-PARSER, a hand-built method that utilizes deep linguistic parsing at extraction time. We find that the learned extractor improves recall by 22% relative to the parser-based approach, while achieving the same level of precision and running at a speed that is 5 times as fast.

3.2.1 Learning an Open Extractor

To measure the precision and recall of both O-NB and O-CRF, we used the random sample of 500 sentences and their manually-tagged tuples described in Section 2.1. The use of this corpus further enables us to examine the system output according to the different relation-independent categories which were previously determined by a human annotator. O-NB and O-CRF were designed and trained prior to obtaining the results of the study; thus the results on this sentence sample provide a fair measurement of their performance.

As shown in Table 3.1, O-CRF extracts relational tuples with a precision of 88.3% and a recall of 45.2%. O-CRF achieves a relative gain in F1 of 63.4% over the O-NB model employed by TEXTRUNNER, which obtains a precision of 86.6% and a recall of 23.2%. The recall of O-CRF nearly doubles that of O-NB.

O-CRF is able to extract instances of the four most frequently observed relation types – Verb, Noun+Prep, Verb+Prep and Infinitive. Three of the four remaining types – Modifier, Coordinate_n and Coordinate_v – which comprise only 8% of the sample, are not handled by TEXTRUNNER due to a simplifying assumption made by TEXTRUNNER that models relationships using only words that occur in positions *between* entity mentions in the sentence. In retrospect, it would be straightforward to extend TEXTRUNNER’s extraction model to consider cues to the left and right of candidate entities in order to find relationships expressed in those contexts.

Punctuation-based relationships are difficult to model with high-precision, and were thus eliminated from the design of TEXTRUNNER’s extraction model. As a result, TEXTRUNNER has difficulty locating appositives, a class of expressions indicating the *IsA*(X, Y) relationship, because they are frequently indicated strictly by punctuation (*e.g.* “*Steve Jobs, the CEO of Apple, ...*” TEXTRUNNER also fails to capture implicit relationships such as that between *Seattle* and *Washington* given the text “*Amazon.com is based in Seattle, Washington.*”

3.2.2 To Parse or Not to Parse

Another question of interest concerns the tradeoffs between parsing at extraction time and using more lightweight forms of text analysis. As discussed in Chapter 2, one species of extractors capable of locating instances of many relations at once is built from two components — a linguistic parser and a set of hand-crafted rules that compose knowledge directly from the parser’s output at extraction time.

To explore the performance of TEXTRUNNER’s lightweight approach to open extraction, we developed a parser-based extraction system which we refer to as O-PARSER. Preferring speed at first, we began with the extraction heuristics previously implemented as part of the DIRT system [54], which used the speedy MINIPAR parser to identify textual paraphrases. Despite its fast performance, we found that the system traded precision for high recall at an undesirable rate. The parser’s grammar-based (*i.e.* non-statistical) nature did not allow us to optimize its output for higher levels of precision.

Deciding to trade speed for accuracy, we then built O-PARSER from Klein and Manning’s Treebank-trained statistical parser and the algorithm described in Figure 3.3 – the same algorithm used by TEXTRUNNER to automatically identify its training data. An important advantage of this design is that the use of this algorithm provides a controlled setting in which to measure how well O-CRF is learning to approximate the knowledge provided by its training data. In addition to using the same parser-to-tuple algorithm, both O-CRF and O-PARSER are trained using the same corpus, the Penn Treebank.

O-PARSER begins by applying the same part-of-speech tagger used by O-CRF to an

input sentence. It then analyses each tagged sentence using Klein and Manning’s parser. The algorithm in Figure 3.3 is then applied; if the algorithm returns a tuple labeled as positive, O-PARSER adds the tuple to its extraction set.

We ran O-PARSER over the set of 500 Web sentences previously used to evaluate O-NB and O-CRF. The results are given in Table 3.1. We observed that the precision of O-CRF and O-PARSER are comparable, with O-PARSER achieving 88.9% precision compared to 88.3% obtained by O-CRF. A significant number of errors made by both systems can be attributed to mistakes made during entity identification. For example O-CRF yields (*Google, confirms, YouTube*) from the sentence, “*Google confirms YouTube acquisition - BBC News* ,” and O-PARSER thinks “*Whereas President Truman*” refers to an object in “*Whereas President Truman established the Presidential Medal of Freedom in 1945,*” . . . While each system uses a different entity-finding mechanism – O-PARSER’s is modeled directly by the parser and O-CRF uses a phrase chunker – we did not detect a significant difference in their behavior.

Some errors made by O-PARSER that are not observed in O-CRF’s output are due to ambiguities concerning phrase attachment, which leads to incorrect extractions such as (*Charlie Chaplin, was born on, London*) from the text “*Charlie Chaplin was born on April 16, 1889 in London . . .*” We hypothesize that O-CRF is able to avoid this problem since the gold-standard parses used to train O-CRF do not contain attachment errors and its lack of a parser at extraction time mitigates the likelihood of this type of mistake.

With respect to recall, we found that O-CRF outperformed O-PARSER with a recall of 45.2% compared to 37.0%. The parser was unable to apply its full statistical model to 102 of 500 sentences and was forced to back-off to a simpler model. As a result, O-PARSER proposed a tuple for only 41.6% of the labeled test instances (208/500), compared to 51.2% (256/500) proposed by O-CRF.

We also compared the time it takes for each method to process a sentence for extraction. On average, sentences in the test corpus contain 21.1 tokens. Using a Pentium 4 3.40GHz with 1GB of memory, O-PARSER was measured to spend an average of 3.2 seconds per sentence. O-CRF is more than 5 times faster, taking only 0.6 seconds per sentence, or an average of 10 seconds per document. Not surprisingly, O-NB which employs a Naive-Bayes classifier, is the fastest of the open extractors we consider, extracting tuples at a rate of

Category	O-NB			O-CRF			O-Parser		
	P	R	F1	P	R	F1	P	R	F1
Verb	100.0	38.6	55.7	93.9	65.1	76.9	96.9	49.7	65.7
Noun+Prep	100.0	9.7	17.5	89.1	36.0	51.3	74.4	25.4	37.9
Verb+Prep	95.2	25.3	40.0	95.2	50.0	65.6	86.5	56.2	68.2
Infinitive	100.0	25.5	40.7	95.7	46.8	62.9	91.7	23.4	37.3
Other	0	0	0	0	0	0	75.0	8.6	15.4
All	86.6	23.2	36.6	88.3	45.2	59.8	88.9	37.0	52.3

Table 3.1: **Open Extraction by Relation Category:** O-CRF outperforms O-NB, obtaining nearly double its recall and increased precision. O-CRF’s gains are partly due to its lower false positive rate for relationships categorized as “Other.”

0.036 seconds per sentence. Thus using 100 machines, it takes less than 2 hours to process 1,000,000 documents with O-NB, 1.15 days with O-CRF, and 5.75 days with O-PARSER.

3.2.3 Conclusion

Our experiment demonstrates that it is possible to learn a relation-independent extractor with high precision, without having to anticipate and thus acquire training data for each relation in the corpus. While unit features are sufficient for high precision open extraction, the use of a Web-based named-entity recognizer such as [29] could further improve the quality of the output. In the next chapter, we present a full Web-scale evaluation of TEXTRUNNER when applied to hundreds of millions of Web pages.

Chapter 4

OPEN EXTRACTION MEETS THE WEB

Information extraction technology has been developed for a variety of text collections, ranging from domain-specific corpora [36, 63, 70, 82] to newspaper articles [1, 78, 99] to the general-purpose Brown Corpus [75]. In recent years, members of the IE community identified Wikipedia as a valuable corpus for knowledge acquisition. As noted by Wu and Weld [96], there are several characteristics of Wikipedia that have made it an increasingly popular target for extraction. The collection of articles offers up-to-date coverage of several million topics and are less prone to factual errors found on arbitrary Web sites (*e.g. Elvis killed President Kennedy*). Wikipedia’s regular page structure makes it easy to separate meaningful textual content from spurious text. Finally, Wikipedia infoboxes, which summarize an object’s key attributes in tabular form, provide a rich source of relational data, while the presence of categories make it possible to assign a taxonomic structure over objects. These observations have sparked several extraction efforts [64, 86, 96] that focus on mining these semi-structured resources as opposed to the unstructured text of Wikipedia articles.

Compared to these extractors, systems such as `TEXTRUNNER`, which process unstructured text, have the potential to acquire a wider range of information. While the number of relations contained in Wikipedia infoboxes is by some measures close to 1,000,¹ many infoboxes are incomplete and many articles lack infoboxes altogether. Figure 4.1 illustrates the difference between the infobox for “*Napoleon Bonaparte*,” and high-ranking extractions found by `TEXTRUNNER`. `TEXTRUNNER` is able to offer information about the emperor’s invasions, battle victories, exile and participation in treaties that is missing from his Wikipedia summary.

Unfortunately for natural language extractors, the importance of solving AI-hard tasks such as anaphora resolution, discourse processing, and inference becomes greater when

¹<http://wiki.dbpedia.org/Ontology>

extracting from a small corpus such as Wikipedia.² For instance, each of these difficult tasks must be solved in order to recognize `CAPITALOF(Paris, France)` from a news article containing the text: “*Paris will be given a new brand image through an advertising campaign based on famous artworks. The artistic side of the French capital will be one of the major aspects of its promotion over the next 12 months.*”

One alternative proposed by the empirical natural language processing community in recent years is to forgo making improvements to deep linguistic analyzers in favor of employing techniques light on natural language understanding over significantly larger datasets such as the Web [9]. By increasing the size and variety of the input corpus, these systems rely on the *redundancy* of the corpus – the existence of multiple, differently phrased statements of the same underlying information. While a fact might be mentioned only once in a news article or biography, it is more likely to be mentioned often, perhaps using easy-to-understand language, on the Web. This observation has been exploited with success in a variety of areas including ambiguity resolution [4, 57, 91], language modeling [47] and thesaurus construction [24].

Redundancy-based techniques for extraction were first explored by the MULDER [51] and ASKMSR [31] question answering systems. These systems performed lightweight text processing, transforming natural language queries (“`What is the capital of France?`”) into search queries that anticipated forms of the answer (“`The capital of France is`” and “`is the capital of France`”) most likely to be present in a large corpus.

The KNOWITALL Web information extraction system [33] also leveraged the redundancy of the Web, relying on the presence of simple sentences that would match domain-independent extraction patterns. The developers of KNOWITALL found that the number of different patterns leading to a given extraction was a good predictor of its quality. The notion that the probability of an extraction increases with the number of distinct sentences in the corpus that suggest the fact became known as the KNOWITALL *hypothesis*. This statement was later formalized by [30] as the URNS model of redundancy and empirically shown to outperform models previously used to assess extraction probabilities. Like KNOW-

²Small, relative to the size of the Web

Napoleon I	
<i>Emperor of the French; King of Italy, Mediator of the Swiss Confederation, Protector of the Confederation of the Rhine</i>	
	
<i>The Emperor Napoleon in His Study at the Tuilleries, by Jacques-Louis David, 1812</i>	
Reign	20 March 1804 – 6 April 1814 1 March 1815 – 22 June 1815
Coronation	2 December 1804
Predecessor	French Consulate (Executive of the French First Republic, with Napoleon as First Consul); Previous ruling Monarch : Louis XVI as King of the French (died 1793)
Successor	Louis XVIII (<i>de facto</i>) Napoleon II (<i>de jure</i>)
Spouse	Joséphine de Beauharnais Marie Louise of Austria
Issue	Napoleon II of France
Full name	Napoleon Bonaparte
Imperial House	Bonaparte
Father	Carlo Buonaparte
Mother	Letizia Ramolino
Born	15 August 1769 Ajaccio, Corsica, France
Died	5 May 1821 (aged 51) Longwood, Saint Helena
Burial	Les Invalides, Paris



TextRunner Search

TextRunner took 4 seconds.

Retrieved **381** results for **napoleon** in argument 1.

Grouping results by argument 1. Group by: [predicate](#) | [argument 2](#)

napoleon - 106 results

Napoleon defeated the Austrians (39), the Prussians (21), the Russians (4), **3 more...**

Napoleon Bonaparte invaded Russia (10), Spain (6), Spain and Portugal (4), **9 more...**

Napoleon Bonaparte was born in Corsica (18), Ajaccio (9), Ajaccio , Corsica (3), Italy (2)

Napoleon was crowned king of King of Italy (28)

Napoleon was exiled to St . Helena (24), Elba (3)

Napoleon Bonaparte conquered Italy (6), Europe (3), France (3), **4 more...**

Napoleon married Josephine de Beauharnais (9), Josephine (2), the Austrian archduchess Marie Louise (2), Jos phine de Beauharnais (2)

Napoleon overthrows the Directory (12), Directorate (2)

Napoleon was defeated in Waterloo (4), Europe (3), Egypt (3), the British (3)

Napoleon abolished the Holy Roman Empire (6), the Meetings (3), the Spanish Inquisition (3)

Napoleon dissolved the Holy Roman Empire (11)

Napoleon wins Battle of Borodino (3), the Battle of Craonne (3), Waterloo (3), the Battle of Vauchamps (2)

Napoleon was defeated at Leipzig (5), Accre (3), British (3)

Napoleon occupied the Netherlands (4), southern Palestine (2), Egypt (2), Rome (2)

Napoleon created Banque de France (3), the Napoleonic Code (3), the French Sanhedrin (2), northern Italy (2)

Napoleon was emperor of France (9)

Napoleon was exiled from France (8)

Napoleon crushed the Prussians (6), the Austrians (2)

Napoleon was sent to St . Helena (8)

Napoleon was proclaimed Emperor (3), the modern Cyrus (3), the French Empire (2)

Napoleon issued the Berlin Decree (5), the Milan Decree (2)

Figure 4.1: A query for “*Napoleon Bonaparte*” illustrates the difference between information found in his Wikipedia summary (top) and facts extracted by TEXTRUNNER (bottom).

ITALL, TEXTRUNNER is a redundancy-based extraction system. TEXTRUNNER assigns a confidence measure to each extracted fact equal to the number of different sentences from which it was found.

Although the size and diversity of Web text make it an attractive source of information, processing the Web also presents a number of challenges. Compared to trusted resources such as Wikipedia and newspaper articles, the Web contains information authored by unreliable or biased sources that may be repeated throughout many documents. Unfortunately, frequency does not always correlate positively with truth. Another problematic characteristic of Web text is that it is often informal. An attempt to extract facts from a blog written in a colloquial nature may unearth many ambiguous or uninteresting assertions, such as (*Aunt Mary, will visit, next July*). Finally, unlike authoritative sources such as Wikipedia and news articles, Web text is more likely to contain grammatical errors that may thwart models of natural language trained from well-formed sentences.

In light of the discussion raised above, we now explore answers to the following questions. Is a redundancy-based design justified for Open IE? What is the difference in the quality, type and size of information extracted from a massive sample of Web pages compared to the knowledge it extracts from Wikipedia, a smaller yet more trustworthy document collection? How many distinct relations does TEXTRUNNER find and what are they like? How does the factbase amassed by the TEXTRUNNER system compare to other recent efforts that have targeted extraction from Wikipedia?

While Section 3.2 assessed the performance of TEXTRUNNER on a small corpus of sentences, this this chapter evaluates TEXTRUNNER when applied to a Web-scale corpus. We first evaluate TEXTRUNNER when applied to 2.5 million Wikipedia articles, and then compare its performance after increasing the size of its input corpus by several orders of magnitude to a total of over 500 million Web pages. The chapter concludes with a discussion of the differences between TEXTRUNNER, a system designed to process arbitrary unstructured English text, to DBPEDIA and YAGO, two large-scale efforts handcrafted to acquire knowledge solely from Wikipedia articles.

The contributions of this chapter are:

- We quantify TEXTRUNNER’s ability to extract instances of previously unspecified relationships from hundreds of millions of Web pages, and demonstrate that TEXTRUNNER automatically discovers millions of facts spanning thousands of different relationships from a large Web corpus. From 500 million Web pages, TEXTRUNNER extracts approximately 218 million facts. Of those, 13.5 million describe more than 16,000 relationships between named entities; the remainder imply abstract properties of general classes.
- We measure the precision and recall of TEXTRUNNER’s extractions from large Web corpora. For a set of 10 well-defined relations commonly studied by the IE community, TEXTRUNNER locates hundreds of thousands of instances with a precision of up to 90.49% from Wikipedia and 92.93% from a corpus that contains Wikipedia plus an additional 500 million pages representative of the general Web.
- We test TEXTRUNNER’s redundancy-driven design by measuring precision as the number of distinct sentences in which an assertion is found increases. Over a vast sample of different relations, assertions extracted from only two different sentences were judged to have a precision of 84.3%. Precision increases to 87.1% when at least 20 different sentences support an extraction, and to 92.3% when 200 distinct supporting sentences are found.
- We discuss the number and quality of relations found by TEXTRUNNER, and compare them to those found in DBPEDIA and YAGO, each of which contains millions of facts extracted from Wikipedia using a combination of manual and semi-automatic methods. TEXTRUNNER finds an order of magnitude more relations than these handcrafted Wikipedia extractors at a comparable level of precision.

4.1 Experimental Setup

4.1.1 Terminology and Parameters

Upon inspection, the type of knowledge extracted by TEXTRUNNER can be classified into two categories. The system locates a large number of *abstract* tuples – tuples that are underspecified, such as (*Einstein, derived, theory*), or imply properties of general classes, such as (*scientist, is author of, article*). TEXTRUNNER also finds what we refer to as *concrete* extractions, where truth of the tuple is grounded in particular entities, for example, (*Einstein, born in, Germany*). While in previous work [5] we reported experiments in which the distinction between abstract and concrete tuples was made by hand, the experiments we report on use part-of-speech tag information about the entities under consideration to automatically characterize TEXTRUNNER’s output.

While abstract assertions are potentially useful for ontology learning, concrete tuples are more useful for IE and question answering tasks. Furthermore, providing human judgments about concrete tuples is significantly easier than abstract tuples; for the most part, they do not require strict judgments involving quantification or disambiguation.³ Finally, algorithms that recognize synonymous objects and relations assume that entities in question refer to real-world, concrete objects. Algorithms for detecting similarity in the context of abstract assertion remains an open problem. For these reasons, we focus on the evaluation of TEXTRUNNER’s ability to extract concrete tuples, from both Wikipedia and the Web in general.

We define a concrete tuple as follows. Given a tuple (X, R, Y) , where X is referred to as the *primary entity* and Y referred to as the *secondary entity*, a concrete tuple has a primary entity that is a proper noun, and the secondary entity that is either a proper noun or date. We use regular expressions and part-of-speech tags to identify these broad classes of entities. We do not use a named-entity tagger to find more specific types such as *Person* or *Location*.

Following extraction, the following distributional constraints are imposed on the entire

³For example, consider the prospect of judging the assertion (*fruit, is low in, fat*) given the existence of the avocado.

set of tuples. We discard tuples in which the predicate R occurs with fewer than n distinct facts in the entire extraction set. Furthermore, a predicate R must be observed with at least e_1 unique primary entities and e_2 unique secondary entities; otherwise all tuples with R are discarded as well. These restrictions are placed due to the belief that tuples containing relations that co-occur with a limited set of entities are either of little interest or are systematic extraction errors. Preferring high precision, perhaps at some cost to recall, we found empirically that setting the above parameters to $n = 50$ and $e_1 = 50$ and $e_2 = 20$ for a corpora containing at least 1 million documents.

4.1.2 Corpora

Open Extraction from Wikipedia

We ran TEXTRUNNER over an English version of Wikipedia archived in January of 2008. At that time, the corpus contained 2,496,172 articles. Given the small size of the collection and the breadth of topics covered, we did not expect the corpus to contain a large amount of redundant information. Thus, contrary to TEXTRUNNER's default design which does not retain tuples found in only one sentence, we permitted TEXTRUNNER to store all extractions when processing Wikipedia.

Of the nearly 32.5 million tuples found by TEXTRUNNER in the Wikipedia corpus, approximately 6.1 million were considered to be concrete according to our definition. After imposing distributional constraints, TEXTRUNNER was found to have extracted 3.8 million concrete assertions about 1.3 million primary named entities using 7657 different relations. After using RESOLVER to identify sets of strings that refer to the same real-world object, TEXTRUNNER found a total of 673 sets containing two or more synonymous relations, but only a handful of sets containing synonymous entities. After merging tuples containing synonymous strings, the final number of distinct relations in Wikipedia as found by TEXTRUNNER was 6742. These results are summarized in Figure 4.2.

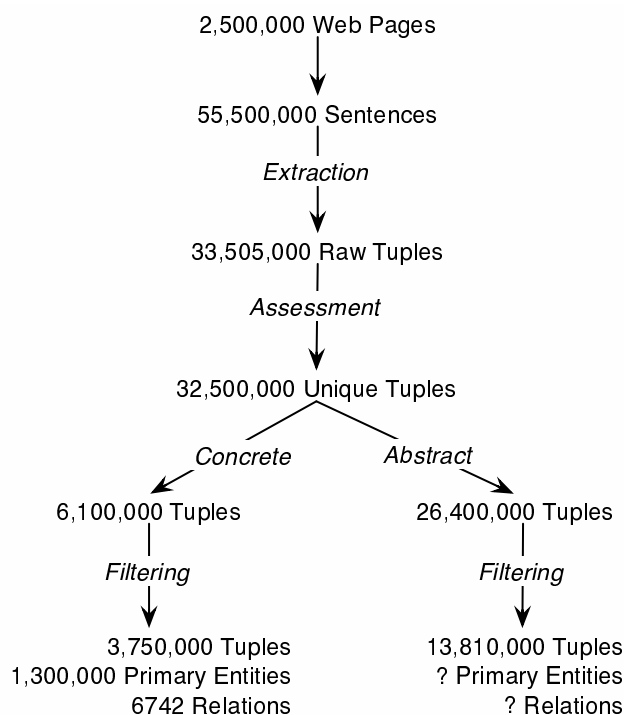


Figure 4.2: **Open Extraction from Wikipedia:** TEXTRUNNER extracts 32.5 million distinct assertions from 2.5 million Wikipedia articles. 6.1 million of these tuples represent concrete relationships between named entities. The ability to automatically detect synonymous facts about abstract entities remains an open problem.

Open Extraction from The General Web

What happens when we augment the size of TEXTRUNNER’s input corpus by several orders of magnitude? In addition to processing Wikipedia, we added 500 million Web pages to the set of documents processed by TEXTRUNNER.⁴ This combination of Wikipedia and the Web is thus referred to as General-Web.

After eliminating extractions found only in a single sentence, TEXTRUNNER was found to extract approximately 850 million raw tuples from General-Web, with 218 million tuples representing unique facts. Of these 218 million, 16.5 million tuples represent concrete facts; 14 million concrete facts remained after applying the aforementioned distributional

⁴The author wishes to thank Google Inc. for providing the corpus.

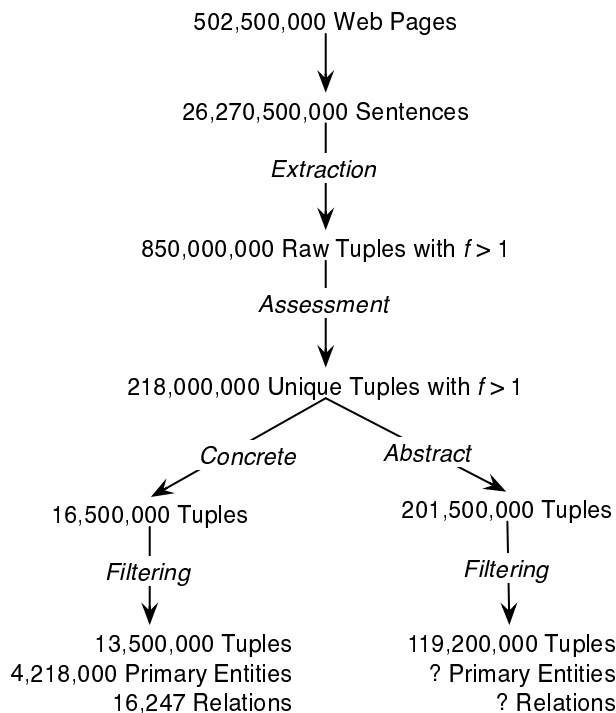


Figure 4.3: **Open Extraction from the Web:** TEXTRUNNER extracts 218 million distinct assertions from 2 or more different sentences in a corpus of over 500 million Web pages (including Wikipedia). 13.5 million of these tuples represent concrete relationships between named entities. Due to TEXTRUNNER’s redundancy-driven design, the measurements do not include assertions extracted from only one sentence. The ability to detect synonyms of abstract entities and relationships represents an area for future work.

constraints to the knowledge base. From this set of assertions, RESOLVER was able to find 582 relation synonym sets and 1671 entity synonym sets, allowing the extraction set to be compressed to 13.5 million tuples. The final set of concrete assertions output by TEXTRUNNER from this corpus spans 16,247 different relations and approximately 4.2 million primary entities. These results are summarized in Figure 4.3.

4.1.3 Evaluation Criteria

Since we cannot measure true recall over a corpus as large as the Web, we use the term *recall* to refer to the size of the set of tuples extracted.

When evaluating the *precision* of extracted tuples, a human assessor uses the following criteria. First, the judge decides whether *the predicate is well-formed*. A predicate R is considered to be well-formed if there is some pair of entities X and Y such that (X, R, Y) is a relation between X and Y . For example $(Wilhelm\ Roentgen, discovered, X-rays)$ contains a well-formed relation, but $(Barack\ Obama, is\ while, Mike\ Huckabee)$ does not.

If a tuple is found to possess a well-formed predicate, it is then judged to see if the entities are appropriate relative to the relation. X and Y are *well-formed entities* for R if X and Y are of a type of entity that can form a relation (X, R, Y) , and if X and Y refer to valid objects (*i.e.*, the proper boundaries have been detected during noun-phrase recognition). Examples of tuples with entities that are not well-formed are $(Whilst\ Shakespeare, died\ on:date, 23\ April)$ and $(Robert\ Louis\ Stevenson, is\ author\ of, Dr.\ Jekyll)$.

Finally, each tuple is judged as *consistent* or not, based on whether it reflects the information contained in the sentence from which it is extracted. For example, $(Al\ Gore, invented, The\ Internet)$ may be tagged as *consistent* due to the presence of sentences such as “*On January 12, Washington Post columnist Al Kamen wrote: “We all know that Al Gore invented the Internet.”*”

In some instances, the validity of assertions suggested by tuples are difficult to disambiguate without the context of the sentence or document from which it is extracted. Examples of such statements are $(Anderson, born\ in:date, 1967)$ and $(Aerospace\ Division, established\ in:date, 1998)$. Other tuples fail to provide complete knowledge due to limitations imposed by the arity of tuples, *e.g.* $(Senator\ Kennedy, asked, Congress)$. For this reason we established both a *strict* and a *non-strict* approach to evaluation. Under a *strict* interpretation, tuples containing ambiguous or incomplete tuples are marked as false. In a *non-strict* context, such tuples are ignored and do not affect the final precision of the judged sample.

4.2 Experimental Results

TEXTRUNNER extracts millions of tuples about thousands of different relations. What is the quality and nature of its output? We begin our evaluation by measuring precision and recall for a set of ten well-studied relations. We then analyze how the redundancy

of the input corpus impacts the precision of `TEXTRUNNER`'s extractions. Finally, while it is difficult to evaluate the quality and recall of the complete set of relations discovered by `TEXTRUNNER`, due to their large number, we provide a look at the relations found by `TEXTRUNNER` relative to other Web-based knowledge bases.

4.2.1 Relation Extraction

We measured precision and recall for ten relations that have often been the focus of study of previously published work in IE, specifically that of `DIPRE` [11], `SNOWBALL` [1], `KNOWITALL` [33], `YAGO` [85, 86] and Bunescu's SSK relation extraction algorithm [12]. While direct comparisons to results previously published by `DIPRE`, `SNOWBALL` and `KNOWITALL` are difficult due to differences in input corpora, we are able to provide a comparison to `YAGO` later in Section 4.2.3 and SSK in Section 5.3. for the appropriate set of relations and data.

A human assessor evaluated a random sample of between 150 and 200 `TEXTRUNNER` extractions, per relation, per input corpus, according to the criteria previously outlined. The results are given in Table 4.1. On average, `TEXTRUNNER` obtains a precision of 92.9% when extracting relations from General-Web, compared to 90.5% when solely processing Wikipedia. The increase was found to be statistically significant at the 90% level, according one-tailed paired t-test [42]. The precision of `TEXTRUNNER`'s relation-independent extractor is comparable to other recent attempts at Web-scale extraction in the single-relation setting. Pasca [62] extracted 1 million instances of the `BORNONDATE` relation from 100 million Web pages with a precision of around 90%. `YAGO` [86] extracted around 350,000 instances of the same relation from Wikipedia with a precision of 93.1% percent.

When strict grading is enforced, where ambiguous tuples are considered incorrect, the precision of the extractions is 87.9% over General-Web and 86.7% over Wikipedia, a non-significant difference. Compared to the general Web, extractions from Wikipedia were ambiguous less often, perhaps due to Wikipedia's more formal, encyclopedic style and the increased presence of conversational text on the Web. For example:

Relation	Wikipedia		General-Web	
	Precision	Recall	Precision	Recall
	Strict/Non-Strict	$f \geq 1$	Strict/Non-Strict	$f \geq 2$
ACQUISITION	81.2/82.2	16,673	84.8/85.3	31,738 ($1.9x$)
BIRTHPLACE	93.9/94.5	122,984	90.1/95.8	225,536 ($1.8x$)
BORNONDATE	90.5/96.8	5,707	89.2/95.7	60,086 ($10.5x$)
CAPITALOF	78.8/78.8	2,621	94.1/94.1	8,514 ($3.2x$)
CEOOF	91.1/91.7	624	78.3/86.1	6,420 ($10.3x$)
DIEDONDATE	87.7/95.5	5,691	90.2/96.3	30,329 ($5.3x$)
ESTABLISHEDONDATE	89.4/95.6	977	92.6/96.5	99,566 ($102.0x$)
HEADQUARTERS	85.4/89.6	3,913	88.6/91.9	52,125 ($13.3x$)
LOCATEDIN	90.5/93.0	63,033	90.6/94.1	159,901 ($2.5x$)
WONPRIZE	76.8/85.6	14,843	80.7/92.4	41,068 ($2.8x$)
Total	86.7 \pm 5.8 / 90.5 \pm 6.2	237,066	87.9 \pm 5.1 / 92.9 \pm 4.1	715,285 ($3x$)

Table 4.1: **Comparison of TextRunner on Wikipedia and General-Web Corpora:** From over 500 million Web pages in the General-Web corpus, TEXTRUNNER extracts instances of ten well-studied relations with an average precision of 92.93%. When applied to the smaller Wikipedia corpus, TEXTRUNNER’s precision is 90.49%, a difference found to be statistically significant with 90% confidence; standard deviation of the mean of differences is 4.0. If tuples containing ambiguous entities are judged as errors, differences in *strict precision* are not significant. The total number of extractions per-relation is also provided, noting that we do not retain singleton extractions ($f = 1$) from the General-Web corpus.

Wikipedia: Steven Paul Jobs (born February 24, 1955) is the co-founder, Chairman and CEO of Apple Inc and former CEO of Pixar Animation Studios.

Web: As many of you know, Steve is the founder & CEO of Apple.

Do I think 'The Steve' is doing a good job as CEO of Apple Computer?

Steve is of course the CEO of Apple Computer and the founder of Pixar.

Considering the errors made by TEXTRUNNER in general, we found that most can be attributed to the presence of an entity that is either malformed (34.4% of errors) or not of the appropriate type (43.1% of errors). The first type of errors are due to difficulties with locating sentence boundaries and noun-phrase boundaries. For instance, (*Rainboworange, based in, IL. Specializing* is mistakenly extracted from “*Rainboworange - Web Design Company based in Chicago, IL. Specializing in . . .*”) and (*Tory Johnson, is ceo of, Women*) from “*Tory Johnson is the founder and CEO of Women for Hire.*” The second type of errors can be largely attributed to errors made by the part-of-speech tagger that cause capitalized words to be mistaken for names, e.g. (*Biography, born in, New York City*) is erroneously extracted from “*Biography - John Leibowitz was born in New York*” The remaining type of errors are extractor-level failures that produce tuples inconsistent with the information contain in the source document from which they are extracted. For example, the extractor proposes the tuple, (*Tirupur, capital of, India*) from the sentence, “*Tirupur is the knitwear capital of India.*”

4.2.2 The Importance of Redundancy

Contrary to Wikipedia, we expect the Web to possess a much higher level of redundancy, due to its size and large number of genres represented. This belief is confirmed by Figure 4.4, which for each corpus, shows the number of concrete assertions that have been extracted a given number of times. Not surprisingly, Wikipedia exhibits a low level of redundancy. An extremely small number of concrete assertions were extracted from more than one distinct sentence – only 44,450 which is only 1.2%. If we discarded all tuples with a count of one when processing Wikipedia, the total number of extractions for the set of ten relations we studied would be too small to be of significant importance. Given the high level of precision of tuples extracted from Wikipedia, most of which were supported by a single sentence, our decision to retain low-frequency extractions when provided with an authoritative corpus appears to be a sound choice. Thus, we conclude that when processing a high-quality, fact-based corpus, the lack of redundancy does not hamper TEXTRUNNER’s effort to extract information with high precision.

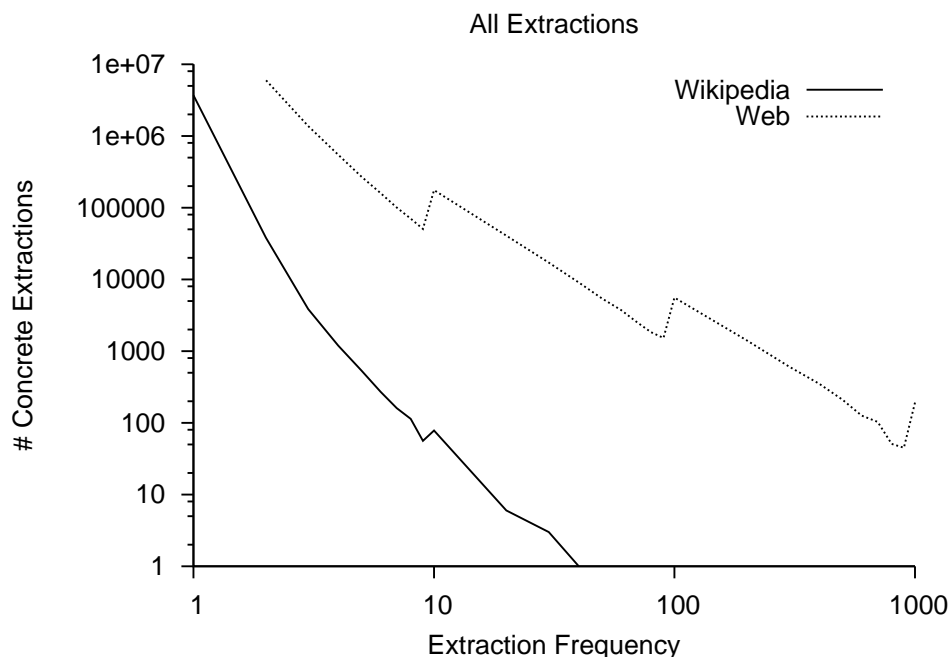


Figure 4.4: **Measuring the Redundancy of Wikipedia and General-Web:** Compared to Wikipedia, where only 1.2% of facts are repeated in multiple sentences, the Web contains many different expressions of the same concept.

To further characterize the nature of the Web, we then found the set of facts extracted from both Wikipedia and General-Web and compared the number of times each fact was found in both corpora. The correlation between extraction frequencies is plotted in Figure 4.5. We found that a fact appearing only once in Wikipedia was found in 4.4 different sentences in our 500-million-page corpus, on average. Given the general reliability of Wikipedia content, this suggests that when considering tuples extracted from a noisier collection of Web documents, tuples found to have an extraction frequency greater than 4 may be more reliable.

Given that General-Web contains a large amount of redundant information, we continued to explore the impact of redundancy on precision relative to this corpus. As previously mentioned, a corpus such as General-Web is likely to contain more ambiguous and unreliable information than Wikipedia. Can TEXTRUNNER, which implements the hypothesis that

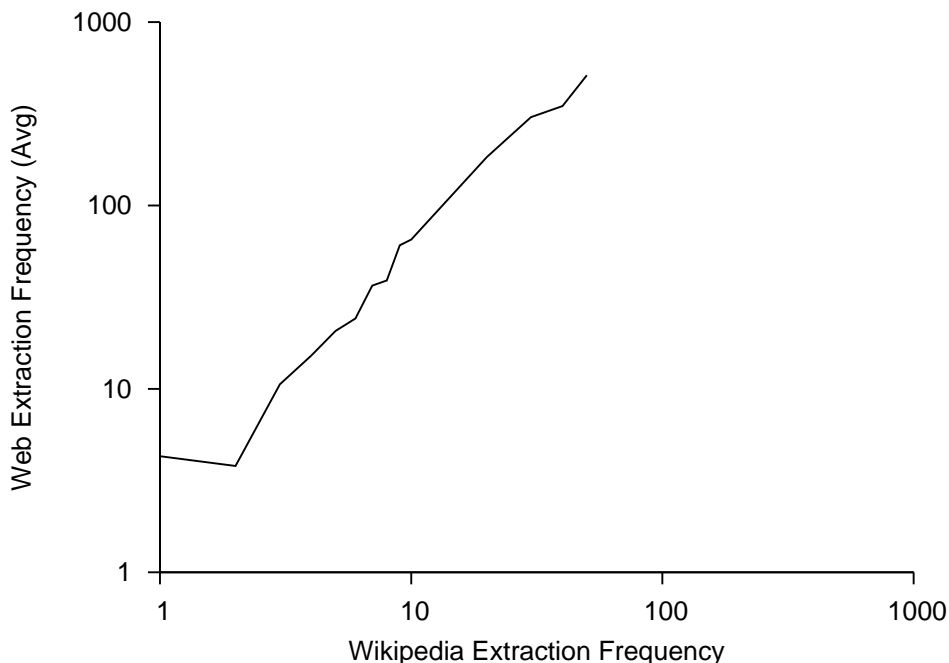


Figure 4.5: A fact found once in Wikipedia was found in more than four different sentences on the Web, on average.

assertions extracted from a variety of sources are more likely to be correct, leverage the presence of redundant information to automatically identify high-quality extractions?

We sampled 850 concrete tuples from TEXTRUNNER’s output from the General-Web corpus. Unlike the previous experiment which considered only instances of pre-specified relations, this sample was collected in a relation-independent manner. Therefore, tuples included in the sample may potentially contain malformed relations. Despite the redundancy of General-Web, the bulk of its tuples are supported by at most 5 different sentences. Therefore, we gave twice as much weight to extractions found within 5 or more sentences in the General-Web corpus. This weighing scheme was applied to increase the likelihood that a meaningful number of high-frequency extractions would be present in our sample.

A human annotator examined the well-formedness and truth of each tuple. Figure 4.6 plots precision P against extraction frequency, f . We found that tuples with $f \geq 2$ were judged to have a precision of 84.3%. At $f \geq 5$, precision jumped to 86.9%, and at $f \geq 50$,

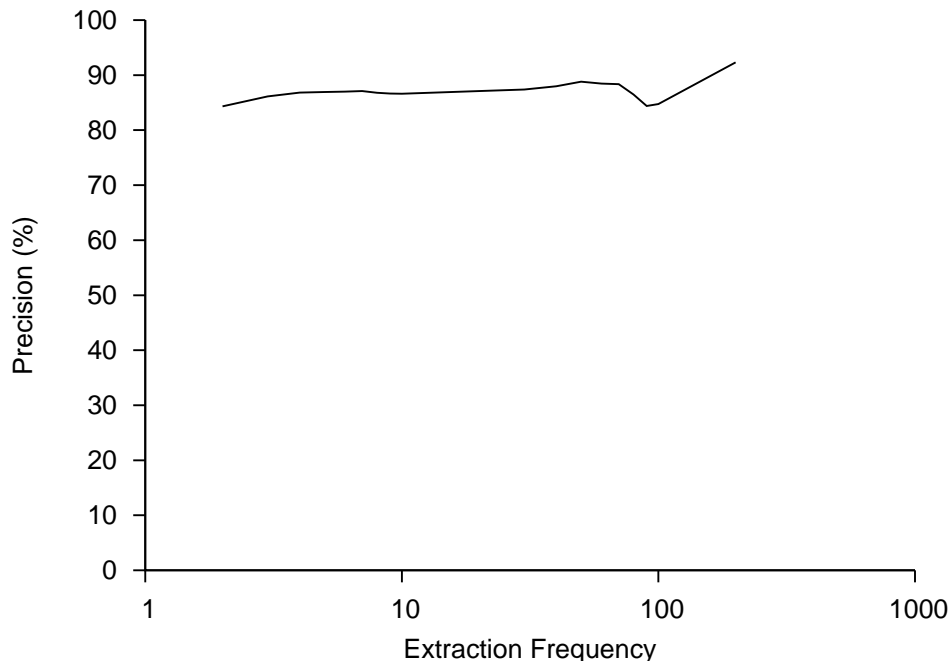


Figure 4.6: **The Impact of Redundancy on Precision:** While assertions found within at most two different sentences had a precision of 84.3%, precision improved to 87.1% when 20 or more supporting sentences were found, and to 92.3% when at least 200 sentences expressed a single assertion.

precision jumped to 88.8%. Compared to per-relation analysis reported in Section 4.2.1, overall precision is slightly lower due to the fact that in the former experiments, the set of tuples evaluated were guaranteed *a priori* to contain a meaningful, well-formed relation.

Based on our sample statistics and with the frequencies computed in Figure 4.4, we can conclude that TEXTRUNNER contains 13.5 million facts at 84% precision ($f \geq 2$), and 1 million facts at 87% precision ($f \geq 5$).

4.2.3 Relation Discovery

Recent efforts at large-scale knowledge acquisition have targeted semi-structured data contained in Wikipedia such as tabular summaries and lists. What is the difference in the type of information that can be gleaned from these resources as opposed to that which

can be found in unstructured text? This section compares the size and quality of relations found by TEXTRUNNER to both DBPEDIA and YAGO, two recent efforts designed to extract structured data from Wikipedia.

DBPEDIA is a community effort designed to extract structured data from Wikipedia [2]. Among other knowledge sources, DBPEDIA includes an ontology that was created by hand from Wikipedia infoboxes. The resulting structure forms a hierarchy over 170 entity types (*e.g. Place, Person, Organization*) describing 882,00 distinct entities. The DBPEDIA ontology contains 940 distinct relationships expressed as RDF, mapped from 2350 variants of popular infobox attributes. (The authors note that despite its impressive size, the ontology covers only a subset of all available attributes.) As of November 2008, the total number of facts is estimated to be near 5.6 million. An example of relations about the entity type *Company* are shown in Figure 4.7.



Figure 4.7: **The DBpedia Ontology:** A hand-built ontology derived from Wikipedia contains 14 relations about objects of type *Company*. 8 additional relations are inherited from its parent type, *Organization*.

Another large source of extracted knowledge is provided by the YAGO IE system, which was previously discussed in Section 2.3. Like DBPEDIA, YAGO also identifies instances of relationships using a set manually developed rules. These rules are designed around the use of a natural language parser and the WordNet lexical database. Instead of performing extraction from the text of articles, YAGO targets both Wikipedia infoboxes and category pages – pages that consist of a list of objects belonging to a class named in title of the page. For example, given that *Napoleon Bonaparte* belongs to a page of “1769 births,”

	TextRunner	DBpedia 3.2	Yago
Entities	4.2 million	882,000	1.5 million
Relations	16,247	940	92
Facts	16.5 million	5.6 million	15 million
Classes	0	170	224,391
Source	Web Text	Wikipedia Infoboxes	Wikipedia Infoboxes and Lists
Architecture	Relation Independent	Relation Specific	Relation Specific

Table 4.2: **A Comparison of Web Extraction Systems:** The estimated number of entities, relations, facts and classes found by each of TEXTRUNNER, the DBPEDIA ontology, and YAGO.

YAGO outputs `BORNONDATE(Napoleon Bonaparte, 1769)`. The set of extraction patterns used by YAGO is both handcrafted and relation-specific. While the expression “`([0 – 9]3, 4) births`” captures instances of `BORNONDATE`, a distinct rule is needed to recognize `ISPOLITICIANOF(Napoleon Bonaparte, France)` given he also falls into the category, “*Emperors of France.*”

As with DBPEDIA, the developers of YAGO designed a mapping by hand that assigns frequently appearing infobox attributes into one of many distinct relations. Specifically, 170 rules were created to map attributes into 92 distinct relations. Of those, 64 represent non-taxonomic relations – `BORNONDATE`, `WONPRIZE`, `HASPOPULATION`, *etc.* – as opposed to hierarchical relationships – `TYPEOF`, `SUBCLASSOF`, `MEANS`, *etc.* These statistics reflect measurements taken in July 2008, as reported in [86]. Table 4.2 summarizes the knowledge contained in the output of TEXTRUNNER, DBPEDIA and YAGO.

While to our knowledge, the accuracy of information contained in DBPEDIA has not been reported, the developers of YAGO measured precision and recall (in terms of the

Relation	YAGO		TEXTRUNNER	
	Precision	# Facts	Precision	#Facts
BORNONDATE	93.14	350,613	95.80	60,086
ESTABLISHEDONDATE	96.84	69,529	96.45	99,566
DIEDONDATE	98.72	168,037	96.32	30,329
WONPRIZE	98.47	13,645	92.41	41,068
Total	96.79	601,824	95.25	231,849

Table 4.3: **TextRunner and Yago:** YAGO utilizes a set of handcrafted patterns to extract data for a small, fixed set of relations from Wikipedia. TEXTRUNNER learns a relation-independent model to extract data from arbitrary Web pages at a comparable level of precision. The number of facts found by the two systems varies by relation and corpus.

number of facts found) for a subset of relations. As summarized in Table 4.3, YAGO’s estimated precision for these relations is comparable to TEXTRUNNER. The number of extracted facts varies by relation, illustrating the tradeoffs between the two approaches. Given a relationship such as BORNONDATE, which are mentioned frequently in Wikipedia using a regular pattern, YAGO’s handcrafted extractor is able to locate more instances than TEXTRUNNER. As we will now discuss, TEXTRUNNER’s advantage lies in its ability to discover a wider variety of relations than can feasibly be covered using YAGO’s labor intensive approach.

It is difficult to evaluate the complete set of relation names discovered by TEXTRUNNER in a standalone manner. Aside from their large number, it is often difficult to make strict decisions about the utility of extracted relations independent of a specific application. For instance, while the tuple (*Senator Robinson, asked, Exxon’s CEO*) may not be a scintillating fact in its own right, it may indirectly provide useful evidence that the politician has some type of relationship with the oil industry.

A look at the relations covered by DBPEDIA, YAGO and TEXTRUNNER highlights the power of Open IE’s relation-independent architecture. To facilitate exploration, we focused on relations describing 7 popular entity types described in DBPEDIA and YAGO. Since

TEXTRUNNER lacks information about entity types, we obtained a list of entities belonging to each of the 7 classes using WordNet. Then for each class, we found all TEXTRUNNER tuples whose primary entity was contained in the list. Using pointwise mutual information (PMI), we computed a ranking of relations relative to each class. PMI, defined here as

$$PMI(R, C) = \frac{Count(e_1 \in C, R, *)}{Count(*, R, *)}$$

measures the association between a relation R and a class C of entities.

Table 4.4 summarizes the number of relations found by each system for each of the 7 types. Additionally, we provide a list of the 25 top relations for 3 domains studied – *Politician* (Table 4.5), *Country* (Table 4.6) and *Company* (Table 4.7) – giving an example for each of three popular top-level categories, *Person*, *Place* and *Organization*. Relations derived by DBPEDIA and YAGO are ordered based on the number of instances. Due to space limitations, we list the relations found by TEXTRUNNER when applied to General-Web, and omit those it finds in Wikipedia alone.⁵

Even when accounting for failures to collapse all distinct forms of a relation into a single predicate – RESOLVER’s recall when attempting to identify relation synonyms was estimated to be around 35% – TEXTRUNNER covers a significantly larger set of relations without any domain-specific tuning. The interestingness of the relations is a subjective decision that is best left to the reader. However, a careful study of the output produced by each system reveals that the set of relations discovered by TEXTRUNNER covers many that are not currently supported by other Web IE systems. While some applications may be interested in finding only definitional attributes that are commonly provided in Wikipedia summaries, others may demand data describing event-based facts, such as a company’s recent product recall, or a country’s attack on a rival nation. TEXTRUNNER enables extraction of both classes of data.

⁵The full set of relations is available online at <http://www.cs.washington.edu/research/textrunner/banko-thesis-data.tar.gz>

Number of Relations				
Entity Type	DBpedia Ontology (Wikipedia)	YAGO (Wikipedia)	TEXTRUNNER (Wikipedia)	TEXTRUNNER (Web)
ACTOR	54	27	822	2569
COMPANY	22	17	314	1478
COUNTRY	30	32	902	3632
POLITICIAN	32	25	300	630
SCIENTIST	32	21	179	283
SOFTWARE	15	5	63	341
UNIVERSITY	31	3	156	384
Total	216	130	2736	9317

Table 4.4: **Estimated Number of Relations about Popular Entity Types:** TEXTRUNNER, an Open IE system finds an order of magnitude more relations than handcrafted Wikipedia extractors DBPEDIA and YAGO.

Domain: Politician		
DBpedia Ontology	Yago	TextRunner (Web)
occupation	bornOnDate	ended
birthPlace	isAffiliatedTo	resigned as leader of
birthDate	bornIn	announced resignation from
deathPlace	diedOnDate	served on committees in
almaMater	hasPredecessor	announced candidacy for
religion	livesIn	has headed
children	hasSuccessor	has run for
nationality	hasWebsite	ran for seat in
residence	diedIn	served as <i>Z</i> with
spouse	graduatedFrom	lost seat to
education	isLeaderOf	to run against
award	isCitizenOf	ran as <i>Z</i> in
party	politicianOf	campaigned in
birthName	actedIn	saw
homeTown	isMarriedTo	ran in:date
title	hasChild	won seat in
relations	isNumber	lost \$ <i>Z</i> to
parents	hasWonPrize	missed of:date
otherNames	created	ran for <i>Z</i> in
knownFor	produced	has been critic of
restingPlace	influences	will run for
successor	wrote	lost election to
predecessor	interestedIn	married <i>Z</i> in:date
ethnicity	isOfGenre	gained title of
deathCause	directed	entered <i>Z</i> as

Table 4.5: **Relations Associated with Politicians:** Using handcrafted rules, DBPEDIA’s ontology contains 32 relations, while YAGO finds 21 relations. TEXTRUNNER finds 630 relations from the general Web in a domain-independent manner. A maximum of 25 results are listed per system.

Domain: Country		
DBpedia Ontology	Yago	TextRunner (Web)
currency	participatedIn	has embassy in
capital	establishedOnDate	has not ratified
language	hasCapital	welcomed
anthem	hasOfficialLanguage	has not signed
governmentType	hasCurrency	gained independence from
latitudeNorthOrSouth	hasPopulation	has rate of
longitudeEastOrWest	hasUTCOffset	established relations with
latitudeDegrees	hasGDP	is ally of
longitudeDegrees	hasCallingCode	acceded to
latitudeMinutes	hasPopulationDensity	walked from
longitudeMinutes	hasTLD	invaded Z in
areaMetroSquareMiles	hasWaterPart	has occupied
populationDensitySquareMiles	hasMotto	recognized independence of
motto	hasHDI	qualified for $\$Z$ in
areaMagnitude	locatedIn	surrendered on:date
demonym	hasWebsite	withdrew in:date
ethnicGroup	dealsWith	will invade
regionalLanguage	hasGini	deported
largestCity	exports	in Z held in
languageType	imports	maintained relations with
largestSettlement	hasNominalGDP	annexed
ethnicGroupsInYear	hasImport	did not attack
elevation	hasInflation	to apologize to
location	hasExpenses	intervened in
coordinates	hasEconomicGrowth	signed treaty of

Table 4.6: **Relations Associated with Countries:** Using handcrafted rules, DBPEDIA’s ontology contains 30 relations, while YAGO finds 32 relations. TEXTRUNNER finds 3632 relations from the general Web in a domain-independent manner. A maximum of 25 results are listed per system.

Domain: Company		
DBpedia Ontology	Yago	TextRunner (Web)
location	established on:date	has shipped
products	has website	should buy
industry	has number of people	started selling
type	created	came out with
revenue	has motto	announce development of
locationCity	has product	has discontinued
parentCompany	produced	has licensed
locationCountry	has successor	released Z for
areaServed	is of genre	to ship
netIncome	lives in	has confirmed to
operatingIncome	has production language	introduced version of
subsid	participated in	announced version of
services	located in	discontinued
owningCompany	created on:date	has announced that
owner	born on:date	has unveiled
divisions	has won prize	has done with
assets		released version of
locations		will be publishing
genre		will release
footnotes		should make
equity		rolled out
language		will be releasing
		unveiled
		acquired Z in:date
		has come out with

Table 4.7: **Relations Associated with Companies:** Using handcrafted rules, DBPEDIA’s ontology contains 22 relations, while YAGO finds 17 relations. TEXTRUNNER finds 1478 relations from the general Web in a domain-independent manner. A maximum of 25 results are listed per system.

Chapter 5

THE RELATIONSHIP BETWEEN OPEN AND TRADITIONAL IE

Traditionally, information extraction has been cast as a classification problem – the task of recognizing whether or not a sentence expresses a given relation among entities of a known type, *e.g.* HEADQUARTERS(COMPANY, CITY). Several approaches have employed support-vector machines tuned with natural language-oriented kernels to classify pairs of entities [13, 21, 99]. Recent progress in the probabilistic inference and machine learning has made it possible to recognize named entities and relations simultaneously [72, 40].

An important distinction between standard IE systems and the new Open IE paradigm is that while traditional extractors can learn to identify instances of relationships using words that appear often in the surrounding context, specific words are not useful indicators of binary relationships *in general*. This relationship is analogous to the relationship between lexicalized and unlexicalized parsers. Statistical parsers are usually *lexicalized*, *i.e.*, they make parsing decisions based on statistics computed over words. However, Klein and Manning [49] showed that *unlexicalized* parsers are more accurate than previously believed, and can be learned in an unsupervised manner. Klein and Manning analyze the tradeoffs between the two approaches to parsing and argue that state-of-the-art parsing will benefit from employing both approaches in concert.

Can the same be said of IE? How does the precision and recall of Open IE compare with a traditional IE system? Is it possible to combine Open IE with a “lexicalized” system to achieve gains in performance? In this chapter, we examine the tradeoffs between relation-specific (“lexicalized”) extraction and relation-independent (“unlexicalized”) extraction and reach a conclusion analogous to that of Klein and Manning.

This chapter addresses the questions raised above and makes the following contributions:

- Section 5.1 compares O-CRF, an extractor developed as part of the TEXTRUNNER Open IE system to a traditional IE system. Without any relation-specific input, O-

CRF obtains the same precision with lower recall compared to a lexicalized extractor trained using hundreds, and sometimes thousands, of labeled examples per relation.

- Section 5.2 presents H-CRF, an ensemble-based extractor that learns to combine the output of the lexicalized and unlexicalized IE systems and achieves a 10% relative increase in precision with comparable recall over traditional IE.
- Section 5.3 introduces R-TEXTRUNNER, an extension to TEXTRUNNER that uses initial knowledge automatically acquired about relations and their instances to achieve additional gains in precision and recall, without any relation-specific labor. At 90% precision, R-TEXTRUNNER improves the recall of Open IE from 16.2% to 47.6%, and locates a relative average of 10% more high-quality facts than a state-of-the-art supervised IE system.

5.1 Traditional Relation Extraction

How does the precision and recall of Open IE compare with that of traditional relation-specific extraction? We compare the behavior of open, or “unlexicalized,” extraction to relation-specific, or “lexicalized” extraction, when a target relation is specified in advance.

To facilitate a controlled comparison, we developed a CRF-based extractor, which we refer as R1-CRF. Although the graphical structure of R1-CRF is the same as O-CRF, R1-CRF differs in a few ways. A relation R is specified *a priori* along with its entity types. R1-CRF is trained from hand-labeled positive and negative instances of R . The extractor is also permitted to use all lexical features, and is not restricted to closed-class words as is O-CRF. Since R is known in advance, R1-CRF outputs a tuple at extraction time, the tuple is believed to be an instance of R . Thus, following extraction with R1-CRF, computing relation synonyms is not necessary.

We compare O-CRF to R1-CRF as opposed to published classification-based relation extraction systems in order to isolate the effects of lexicalization *vs.* unlexicalization, supervised *vs.* self-supervised training, and single-relation *vs.* multi-relation extraction, keeping the model-structure constant.

5.1.1 Experimental Results

To compare performance of the extractors when a small set of target relationships is known in advance, we used labeled data for four different relations – corporate acquisitions, birthplaces, inventors of products and award winners. The first two datasets were collected from the Web, and made available by Bunescu and Mooney [12]. To augment the size of our corpus, we used the same technique to collect data for two additional relations, and manually labeled positive and negative instances by hand over all collections. While in the previous chapter, the size of the Web corpus made it infeasible to measure absolute recall, the use of this hand-labeled dataset makes it possible to estimate recall according to the traditional definition – the proportion of positive instances that the system extracts for each relation. For each of the four relations in our collection, we trained R1-CRF from labeled training data, and ran each of R1-CRF and O-CRF over the respective test sets, and compared the precision and recall of all tuples output by each system. Table 5.1 provides the amount of data collected for each relation.

	Train	Test
Relation	Examples	Examples
Acquisition	3042 (1481)	1017 (169)
Birthplace	1852 (260)	601 (45)
InventorOf	1000 (421)	300 (120)
WonPrize	1000 (255)	390 (216)

Table 5.1: **Summary of IE Corpora:** A collection of labeled Web sentences containing instances of four relations. The numbers in parenthesis indicate the number of positive examples.

Figure 5.1 shows that without any relation-specific data, O-CRF can achieve a level of precision comparable to R1-CRF. However the recall of the open extractor is lower; at 90% precision, O-CRF obtains 13.6% recall compared to 35.1% found by R1-CRF. At 75% precision, O-CRF’s recall is 18.4% compared to R1-CRF’s 58.4%,

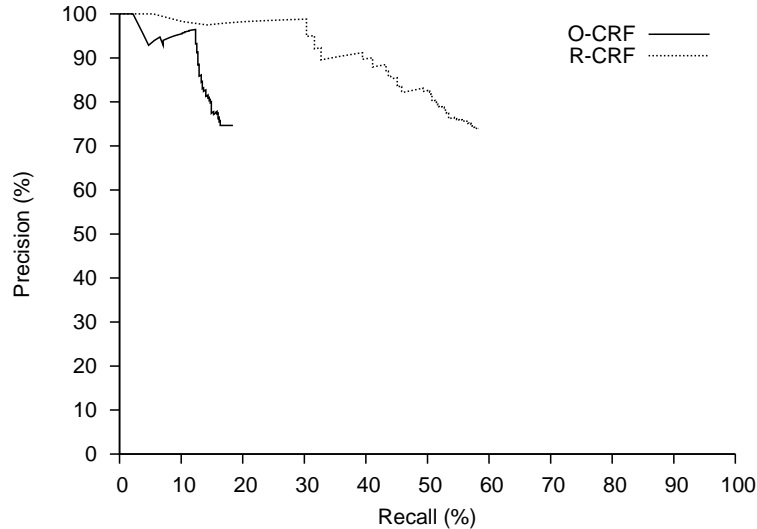


Figure 5.1: **Precision-Recall Tradeoff Between Open and Traditional IE:** O-CRF can achieve high precision without any relation-specific input. Recall is lower than R1-CRF, a traditional extractor trained using thousands of examples per relation.

A large number of false negatives on the part of O-CRF can be attributed to its lack of lexical features, which are often crucial when part-of-speech tagging errors are present. For instance, in the sentence, “*Yahoo To Acquire Inktomi*”, “*Acquire*” is mistaken for a proper noun, and sufficient evidence of the existence of a relationship is absent. The lexicalized R1-CRF extractor is able to recover from this error; the presence of the word “*Acquire*” is enough to recognize the positive instance, despite the incorrect part-of-speech tag.

Another source of recall issues facing O-CRF is its ability to discover synonyms for a given relation. We found that while RESOLVER improves the relative recall of O-CRF by nearly 50%, O-CRF locates fewer synonyms per relation compared to its lexicalized counterpart. With RESOLVER, O-CRF finds an average of 6.5 synonyms per relation compared to R1-CRF’s 16.25.

Exactly how many training examples per relation does it take R1-CRF to achieve a comparable level of precision? We varied the number of training examples given to R1-CRF, and found that in 3 out of 4 cases it takes hundreds, if not thousands of labeled

Relation	O-CRF		R1-CRF		
	P	R	P	R	Train Ex
Acquisition	75.6	19.5	67.6	69.2	3042*
Birthplace	90.6	31.1	92.3	53.3	600
InventorOf	88.0	17.5	85.7	27.8	1000*
WonPrize	62.5	15.3	62.7	24.1	35
All	75.0	18.4	75.0	43.8	>4677

Table 5.2: **An Effort to Match Open IE:** For 4 relations, a minimum of 4677 hand-tagged examples is needed for R1-CRF to approximately match the precision of O-CRF for each relation. A “*” indicates the use of all available training data; in these cases, R1-CRF was unable to match the precision of O-CRF.

examples for R1-CRF to achieve precision comparable to O-CRF. In two cases – acquisitions and inventions – R1-CRF is unable to match the precision of O-CRF, even with many labeled examples. Table 5.2 summarizes these findings.

In light of our findings, the relative tradeoffs of open versus traditional IE are as follows. Open IE automatically offers a high level of precision without requiring manual labor per relation, at the expense of recall. When relationships in a corpus are not known, or their number is massive, Open IE is essential. When higher levels of recall are desirable for a small set of known target relations, traditional IE is more appropriate. However, in this case, one must be willing to undertake the cost of acquiring labeled training data for each relation, via human annotators or a computational procedure such as bootstrapped learning.

5.2 Hybrid Relation Extraction

Since O-CRF and R1-CRF have complementary views of the extraction process, it is natural to wonder whether they can be combined to produce a more powerful extractor. *Empirical multistrategy learning* refers to the combination of multiple learning approaches using a single algorithm [28]. In a variety of machine learning settings, the use of an ensemble of diverse classifiers during prediction has been observed to yield higher levels of performance

compared to individual algorithms. We now describe an ensemble-based or *hybrid* approach to RE that leverages the different views offered by open, self-supervised extraction in O-CRF, and lexicalized, supervised extraction in R1-CRF.

5.2.1 Stacked Relation Extraction

Stacked generalization, or *stacking*, [94], is an ensemble-based framework in which the goal is learn a meta-classifier from the output of several base-level classifiers. The training set used to train the meta-classifier is generated using a leave-one-out procedure: for each base-level algorithm, a classifier is trained from all but one training example and then used to generate a prediction for the left-out example. The meta-classifier is trained using the predictions of the base-level classifiers as features, and the true label as given by the training data.

Previous studies [88, 100, 79] have shown that the probabilities of each class value as estimated by each base-level algorithm are more effective features when training meta-learners. Stacking was shown to be consistently more effective than voting, another popular ensemble-based method in which the outputs of the base-classifiers are combined either through majority vote or by taking the class value with the highest average probability.

Other researchers have found the stacking framework to yield benefits in the context of IE. Freitag [35] used linear regression to model the relationship between the confidence of several inductive learning algorithms – rote learning, Naive Bayes, grammatical inference and a relational rule learning – and the probability that a prediction is correct. Over three different document collections, the combined method yielded improvements over the best individual learner for all but one relation. The efficacy of ensemble-based methods for extraction was further investigated by [79], who experimented with combining the outputs of a rule-based learner, a Hidden Markov Model and a wrapper-induction algorithm in five different domains. Of a variety of ensemble-based methods, stacking proved to consistently outperform the best base-level system, obtaining more precise results at the cost of somewhat lower recall.

We used the stacking methodology to build an ensemble-based extractor, referred to as H-CRF. Treating the output of an O-CRF and R1-CRF as black boxes, H-CRF learns to

Relation	R-CRF			Hybrid		
	P	R	F1	P	R	F1
Acquisition	67.6	69.2	68.4	76.0	67.5	71.5
Birthplace	93.6	64.4	76.3	96.5	62.2	75.6
InventorOf	85.7	50.0	42.0	87.5	52.5	65.6
WonPrize	73.6	52.8	61.5	75.0	50.0	60.0
All	74.6	58.2	65.4	79.2	56.9	66.2

Table 5.3: **Hybrid Information Extraction:** A hybrid extractor that uses Open IE improves precision for all relations, at an insignificant cost to recall.

predict which, if any, tokens found between mentions of a pair of entities (e_1, e_2) , indicates a relationship. Due to the sequential nature of our RE task, H-CRF employs a CRF as the meta-learner, as opposed to a decision tree or regression-based classifier.

H-CRF uses the probability distribution over the set of possible labels according to each O-CRF and R1-CRF as features. To obtain the probability at each position of a linear-chain CRF, the constrained forward-backward technique described in [22] is used. H-CRF also uses a numeric string-similarity feature that compares the similarity of the relations predicted by O-CRF and R1-CRF and a numeric feature that indicates whether either or both base extractors return “no relation” for a given pair of entities. Finally, at each given position i between e_1 and e_2 , H-CRF uses the presence of the word observed at i as a feature, as well as the presence of the part-of-speech-tag at i .

5.2.2 Experimental Results

In this section, we evaluate the performance of H-CRF, an ensemble-based extractor that learns to perform RE for a set of known relations based on the individual behaviors of O-CRF and R1-CRF.

As shown in Table 5.3, the use of Open IE as part of H-CRF, improves precision from 74.6% to 79.2% with only a slight decrease in recall. Overall, F1 improved from 65.4% to

66.2%. However, one disadvantage of a stacking-based hybrid system is that labeled training data is still required.

5.3 *The R-TextRunner System*

In Section 5.1.1 we found that while our self-supervised Open IE system could match the precision of a supervised IE system, its recall was notably lower. Thus, it appeared that the developer of an IE system might be forced to make a tradeoff: develop an Open IE system, incurring only a one-time development cost to extract instances of a massive set of unknown relationships, but accept low recall, or invest the time and expertise required to enumerate a set of possibly interesting relationships from a large body of text, and then train an extractor for each one.

Perhaps, however, we do not need to be resigned to making such a decision. Earlier in Section 1.2.1, we surveyed weakly-supervised IE, a class of systems that made significant gains in automating the development process. Compared to supervised algorithms, in which *every* example in a training set must be labeled as positive or negative, weakly-supervised algorithms demand only a small set of handcrafted seed inputs and an unlabeled training corpus. For IE, seeds take the form of extraction patterns or entities known to be positive or negative instances of the relation. Two recent Web IE systems [12, 62], demonstrated the ability to extract relations in the weakly-supervised setting using only 8-10 training examples per relation. These approaches have yielded significant reductions in manual labor relative to supervised algorithms which typically require several orders of magnitude more labeled inputs. However, when the goal is to extract instances for a large set of relations, *i.e.* the 25,000 verbs defined in WordNet or the 8,000 different attributes that appear in Wikipedia summaries, even this amount of effort becomes non-trivial.

Recognizing that seed inputs need not necessarily be specified by hand, two self-supervised IE systems reduced the labor required for system development even further. A baseline version of the KNOWITALL system [33] used strictly domain-independent patterns to extract instances of a given relationship. While recall was sometimes low, KNOWITALL's relation-independent patterns were successful at identifying facts with very high levels of accuracy. Thus, a subsequent version of KNOWITALL used high-scoring output from the baseline sys-

tem as seed inputs to a pattern learning algorithm. KNOWITALL’s acquisition of relation-specific extraction patterns yielded a 50% to 80% boost in recall and 28% to 35% reduction in error. URES [71] also leveraged high-quality output from baseline KNOWITALL to train a pattern-learning algorithm that increased the number of facts extracted while maintaining the same precision of the baseline system.

Wu, Hoffmann and Weld [95] also demonstrated they could improve the recall of a high-precision extraction algorithm by applying shrinkage, a technique that enables a system to find additional training examples within a corpus when the initial set of training instances is sparse. Additional gains in recall were observed after retraining the extractors with a corpus augmented with both the output of TEXTRUNNER and additional examples automatically identified from a large Web corpus.

Is it possible to employ a similar approach in the open extraction setting? Once TEXTRUNNER has initially discovered relationships present within a corpus and located a set of high-quality instances, can we learn to improve extraction of specific relations of interest? Inspired by KNOWITALL and URES, we now describe R-TEXTRUNNER, a system that uses extractions deemed trustworthy by TEXTRUNNER’s Assessor to automatically identify a set of sentences containing positive and negative instances of a given relation. The labeled data is used to learn more about individual relationships using a traditional relation extraction algorithm. Our experiments demonstrate that we can leverage TEXTRUNNER’s knowledge about entities and relationships to improve extraction recall on a per-relation basis, *without incurring any additional manual labor*.

5.3.1 Algorithm

Instead of requiring a set of seed patterns or named entity pairs, the R-TEXTRUNNER system uses tuples automatically found by an open extraction system to seed a self-supervised learning process about specific relations. While this process is relation-specific, no additional human labor is necessary. The relation names and training data are provided automatically by the Open IE system. Specific relationships can be targeted using a data-driven approach (*e.g.* those relationships about which the system currently knows most/least about) or a

user-driven approach (*e.g.* the relationships most often queried or those explicitly demanded by a user). The automatically labeled data can be used to train any traditional relation extraction algorithm.

The process begins by running TEXTRUNNER over an unlabeled corpus \mathcal{C} , which produces a set of named relationships \mathcal{R} . For every $R \in \mathcal{R}$, TEXTRUNNER also outputs T_r^+ , the set of tuples believed to be positive instances of R . While the accuracy of TEXTRUNNER’s extractor is sufficiently high, it is likely that T_r^+ will contain some false positives. We can mitigate this issue by selecting only those instances extracted from a large number of sentences, which are more likely to be trustworthy.

While obtaining a reliable set of positive examples is rather straightforward, how can we choose a set of negative training examples for a given relation R ? Our solution is based on the observation that in many cases, more than one relation is observed between a pair of objects e_1 and e_2 . While TEXTRUNNER identifies that *(Mozart, was born in, Vienna)* satisfies the *Birthplace* relation, it also finds that *(Mozart, worked in, Vienna)* and *(Mozart, left, Vienna)* are asserted in the corpus. Due to RESOLVER, the system already knows that “*worked in*” and “*left*” are not synonymous with “*was born in.*” R-TEXTRUNNER uses precisely this knowledge to automatically label sentences in which a relationship is extracted between entities in the seed tuples. R-TEXTRUNNER uses sentences yielding tuples of the form (e_1, P, e_2) where $P \neq R$ and (e_1, R, e_2) also holds as negative examples.

Perhaps greater than the potential of false positives, is the danger of false negatives. TEXTRUNNER’s knowledge of relation synonyms is imperfect. For instance, RESOLVER did not find that “*X is from Y*” can be a synonym of “*X was born in Y.*” By default, TEXTRUNNER parameterizes RESOLVER to favor high precision resolution of synonyms, making potential sacrifices in recall. To use RESOLVER for self-supervised training, we find that it helps significantly to recompute relation synonyms on demand using a more relaxed parameter setting.

Specifically, R-TEXTRUNNER operates as follows:

1. Run TEXTRUNNER on \mathcal{C} , an unlabeled corpus.
2. For a relation $R \in \mathcal{R}$, the set of relations discovered by TEXTRUNNER:

- (a) Get T_r^+ , the set of tuples asserting $t = (e_1, R, e_2)$ discarding any t found in fewer than m sentences in \mathcal{C} . Retain the sentences from which each tuple was extracted.
- (b) Get T_r^- , the set of tuples asserting $t = (e_1, P, e_2)$ such that $P \neq R$ and (e_1, R, e_2) also holds. Retain the sentences from which each tuple was extracted.
- (c) Let $T = T_r^+ \cup T_r^-$. Run RESOLVER with input T . Output the hypothesized synonyms of R .
- (d) For each $t = (e_1, x, e_2) \in T$, label each sentence asserting t as positive if x is a synonym of R and negative otherwise.
- (e) Use the set of labeled sentences to train an extractor for R .

5.3.2 Experimental Results

In this section we compare several paradigms for IE using the following embodiments:

- **Open IE:** Open extractor that discovers and extracts a large set relations at once from an unlabeled corpus. We use the TEXTRUNNER system as described in Chapter 3.
- **Supervised IE:** Supervised extractor trained one relation at a time from a fully labeled corpus. We use a public implementation of Bunescu’s Subsequence Kernel (SSK) algorithm¹ [13]. SSK uses a support vector machine to model relations between a set of entities. Using sequences of features derived from words, part-of-speech tags and phrase chunks, SSK was shown to outperform a similar kernel-based algorithm that used features derived from a dependency parser.
- **Weakly-Supervised IE:** Weakly-supervised extractor trained one relation at a time from a handful of inputs and an unlabeled corpus. We use published test data for SSK-MIL [12], an extension of SSK that belongs to the class of multiple instance learning (MIL) algorithms. MIL [87] is a weakly supervised learning framework in which a model is trained from bags of examples. As opposed to individual instance labels, only the bag labels are required for learning. A bag is considered positive if

¹http://ace.cs.ohiou.edu/~razvan/code/ssk_cor.tar.gz

it contains at least one positive example; thus, one only needs to provide a pair of entities known to embody the relation along with an unlabeled corpus sufficiently large enough to guarantee the presence of a positive instance. Relative to a Web IE corpus containing thousands of training examples per relation, SSK-MIL is competitive with its fully-supervised counterpart. At 90% precision, the recall of SSK and SSK-MIL is 57.8% and 44.3%, respectively.

- **Self-Supervised IE:** The R-TEXTRUNNER algorithm self-trained one relation at a time using automatically generated relation names and training data from TEXTRUNNER. R-TEXTRUNNER employs the SSK algorithm to train each new extractor.

To evaluate each paradigm, we used the training and test corpora described in Section 5.1.1 which covers four relations — *Acquisition*, *Birthplace*, *InventorOf* and *WonPrize*. SSK trains each extractor using all available training examples: *Acquisition* (3042), *Birthplace* (1852), *InventorOf* (1000), *WonPrize* (1000).

SSK-MIL uses 8 seed inputs per relation to label an unlabeled version of the same training corpus. We had access only to the algorithm’s output as reported on the original test relations, *Acquisition* and *Birthplace*, and not to the system implementation. Therefore, we are not able to report performance of the weakly-supervised algorithm for *InventorOf* and *WonPrize*, the two relations whose data we added to the collection.

R-TEXTRUNNER uses seed data output by TEXTRUNNER after it has been applied to a corpus of 500 million Web pages previously described in Section 4.1.2. Tuples appearing fewer than 3 times in the corpus were not used as seeds. The baseline version of R-TEXTRUNNER was constrained to use only the same number of training examples per relation as SSK and SSK-MIL. The training sentences were chosen randomly from all that were available from TEXTRUNNER while maintaining the balance of positive and negatives in the full set. The full version of R-TEXTRUNNER was permitted to use 10,000 randomly-chosen training examples per relation. While more examples were available, we found that changes in F-measure, the harmonic mean of precision and recall, did not vary significantly after the 7,500-10,000 mark.

Figure 5.2 compares TEXTRUNNER, our baseline Open IE system, to its extension R-TEXTRUNNER. The precision-recall curves demonstrate the utility of self-supervised learning. Using only 1,000 self-labeled examples per relation, R-TEXTRUNNER finds 2.3 the number of correct instances found by TEXTRUNNER. At 90% precision, recall improves from 16.2% to 37.0% over the set of 4 test relations. Significant gains are made when R-TEXTRUNNER uses an order of magnitude more Web training data, which is labeled at no additional expense. Using 10,000 self-labeled examples, recall increases by nearly a factor of three. At 90% precision, R-TEXTRUNNER improves the recall of Open IE from 16.2% to 47.6%. On a Pentium 4 3.40GHz with 1GB of memory, training the classifier takes approximately 10 minutes with 1,000 examples, and 3 hours with 10,000 examples.

Table 5.4 provides a list of expressions R-TEXTRUNNER believes to be positive indicators for each test relation. Not surprisingly, there is some noise introduced by the algorithm’s self-labeling process. For example, in the case of the *WonPrize* relation, “*leave*” is mistakenly identified as a synonym, as is “*win while*” which is a malformed relation, “*be awarded*” is erroneously tagged as a negative example. Even with approximately 5% of instances mistakenly labeled, R-TEXTRUNNER is able to improve recall without making sacrifices to precision.

Figure 5.3 plots the precision and recall of TEXTRUNNER and R-TEXTRUNNER relative to supervised and weakly-supervised IE systems. Without any relation-specific labor or hand-tagging of data, R-TEXTRUNNER locates a relative average of 10% more high-quality facts than a state-of-the-art supervised IE system. We also computed area under the precision-recall curve (AUC) in the high-precision range (90% - 100%) by measuring the trapezoidal areas created between each point. On average, R-TEXTRUNNER reduces the error rate of a supervised extractor by 4.5%, increasing area under the curve from 0.430 to 0.474. For each relation, the difference is statistically significant at $p < 0.01$ according to a two-sample t-test, using the methodology for measuring the standard deviation of AUC given in [68].

Table 5.5 summarizes the paradigms studied. At a one-time cost incurred per language of interest, Open IE can locate high-quality instances of a large set of relationships whose identity need not be known in advance. While recall may at first be lower than traditional

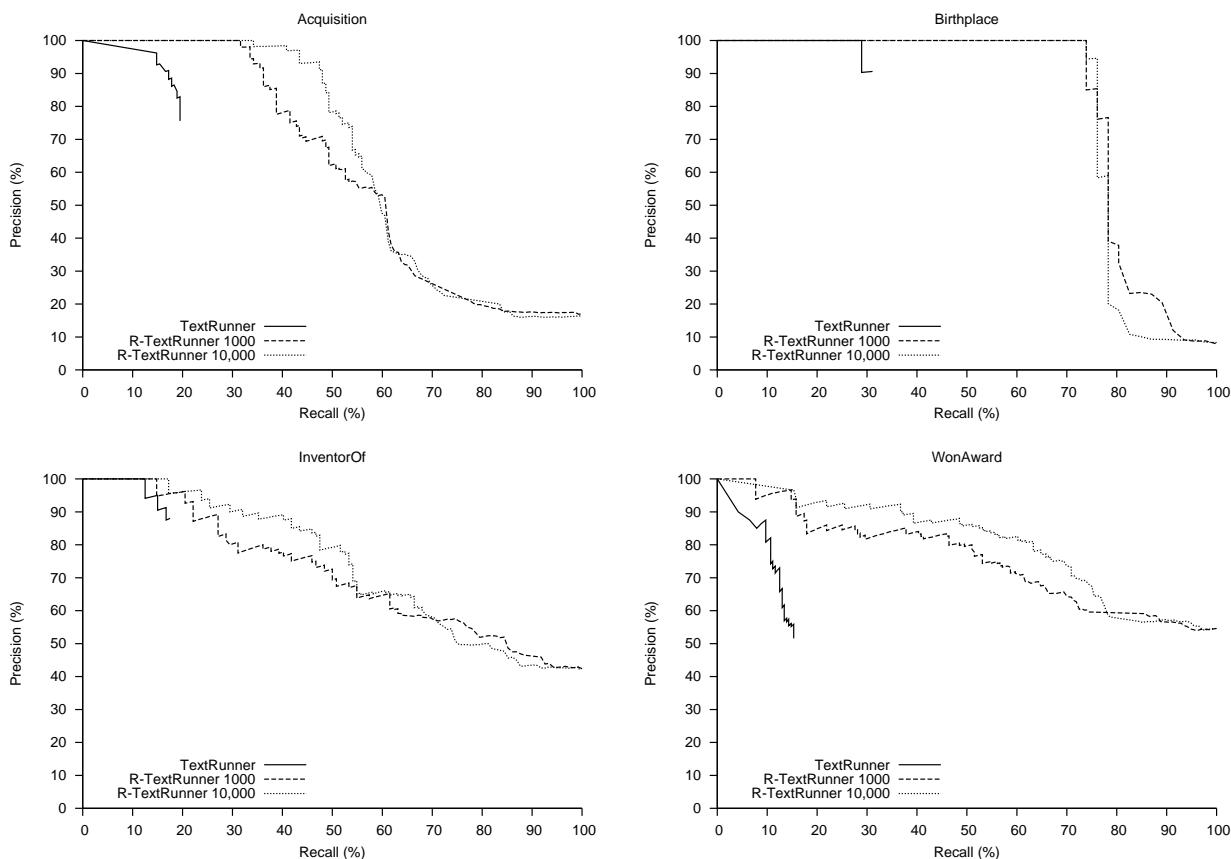


Figure 5.2: **Automatically Improving the Recall of Open IE:** Once a set of relation names and reliable instances have been extracted by TEXTRUNNER, R-TEXTRUNNER uses them to improve the recall on a per-relation basis. On average, recall more than doubles when using 1,000 examples per relation and triples when using 10,000 examples.

supervised extractors, one can leverage the knowledge initially acquired by Open IE to bootstrap additional learning about specific relations as needed. Thus, both state-of-the-art precision and recall can be achieved without making labor-intensive or domain-specific efforts.

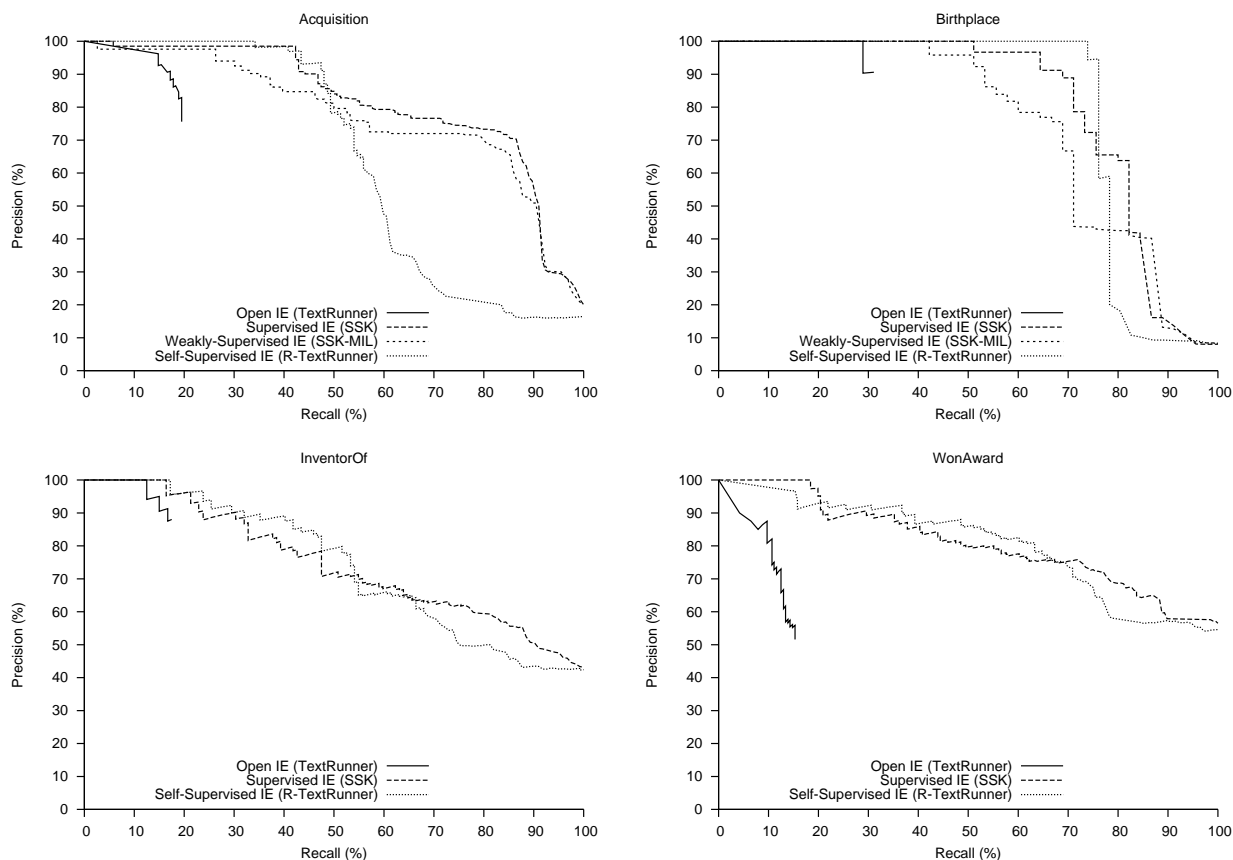


Figure 5.3: **State-of-the-Art Web IE without the Effort:** For a set of 4 relations, R-TEXTRUNNER, a self-supervised Web IE system, uses the knowledge acquired by an Open IE system to automatically improve its performance. Without any manual input, R-TEXTRUNNER identifies a larger number of high-precision assertions than its supervised and weakly-supervised counterparts, which require anywhere from tens to thousands of labeled inputs per relation.

Acquisition	Birthplace	InventorOf	WonAward
<i>X.* acquire .*Y</i>	<i>X.* be born in .*Y</i>	<i>X.* invent .*Y</i>	<i>X.* win .*Y</i>
<i>X.* to acquire .*Y</i>	<i>X.* would be born in .*Y</i>	<i>X.* create .*Y</i>	<i>X.* have win .*Y</i>
<i>X.* to buy .*Y</i>	<i>X.* have be born in .*Y</i>	<i>X.* develop .*Y</i>	<i>X.* will win .*Y</i>
<i>X.* have acquire .*Y</i>	<i>X.* be born in of .*Y</i>	<i>X.* have invent .*Y</i>	<i>X.* to win .*Y</i>
<i>X.* purchase .*Y</i>	<i>X.* who be born in .*Y</i>	<i>X.* who invent .*Y</i>	<i>X.* receive .*Y</i>
<i>X.* have buy .*Y</i>	<i>X.* to be born in .*Y</i>	<i>X.* introduce .*Y</i>	<i>X.* win in .*Y</i>
<i>X.* is buying .*Y</i>	<i>X.* will be born in .*Y</i>	<i>X.* patent .*Y</i>	<i>X.* can win .*Y</i>
<i>X.* merge with .*Y</i>	<i>X.* marry be born in .*Y</i>	<i>X.* is inventing .*Y</i>	<i>X.* leave .*Y</i>
<i>X.* have purchase .*Y</i>	<i>X.* be born in before .*Y</i>	<i>X.* demonstrate .*Y</i>	<i>X.* would win .*Y</i>
<i>X.* will acquire .*Y</i>	<i>X.* must be born in .*Y</i>	<i>X.* bring .*Y</i>	<i>X.* is winning .*Y</i>
<i>X.* buy out .*Y</i>		<i>X.* have create .*Y</i>	<i>X.* may have win .*Y</i>
<i>X.* is acquiring .*Y</i>		<i>X.* may have invent .*Y</i>	<i>X.* could win .*Y</i>
<i>X.* pay for .*Y</i>		<i>X.* have bring .*Y</i>	<i>X.* will win in .*Y</i>
<i>X.* will buy .*Y</i>		<i>X.* can create .*Y</i>	<i>X.* have win in .*Y</i>
<i>X.* offer for .*Y</i>		<i>X.* who develop .*Y</i>	<i>X.* win while .*Y</i>
<i>X.* snap up .*Y</i>		<i>X.* is creating .*Y</i>	<i>X.* win with .*Y</i>
<i>X.* offer .*Y</i>		<i>X.* is introducing .*Y</i>	
<i>X.* which acquire .*Y</i>			
<i>X.* which own .*Y</i>			
<i>X.* invest in .*Y</i>			
<i>X.* which buy .*Y</i>			
<i>X.* aquire .*Y</i>			
<i>X.* take over .*Y</i>			
<i>X.* pick up .*Y</i>			
<i>X.* bring .*Y</i>			

Table 5.4: **R-TextRunner**: Once TEXTRUNNER has discovered a set of relations and trustworthy instances of each, the output can be used to automate additional learning about a particular relation. For the four relations we studied, we provide the phrases most-frequently used by R-TEXTRUNNER to automatically label positive instances for learning. Words have been normalized to their base forms.

	Open	Supervised	Weakly-Supervised	Self-Supervised
Relation	AUC $P \geq 90\%$			
Acquisition	0.170 ± 0.041	0.458 ± 0.006	0.340 ± 0.006	0.475 ± 0.007
Birthplace	0.309 ± 0.033	0.681 ± 0.020	0.528 ± 0.021	0.760 ± 0.021
InventorOf	0.164 ± 0.011	0.293 ± 0.009	<i>n/a</i>	0.310 ± 0.011
WonPrize	0.040 ± 0.038	0.287 ± 0.006	<i>n/a</i>	0.352 ± 0.007
Labor Cost				
	$O(1)$	$O(1000R)$	$O(10R)$	$O(1)$
Runtime				
	$O(kD)$	$O(RD)$	$O(RD)$	$O(kD + RD)$

Table 5.5: **A Comparison of IE Paradigms:** Given a corpus of D documents and R relations to extract, we compare an open extractor, a supervised extractor, a weakly-supervised extractor and a self-supervised extractor trained using the open extractor and the Web. Listed are area under the precision/recall curve (AUC) in the high-precision range (90%-100%). With the exception of the open extractor, each system uses the same underlying extraction algorithm to learn a traditional single-relation extractor. While the labor cost of supervised and weakly-supervised learning is on the order of 10 and 1000 hand-labeled instances per relation, respectively, the amount of manual effort required by the open and self-supervised systems is independent of R . The self-supervised extractor consistently locates a greater number of high-precision extractions than the completely supervised version.

Chapter 6

CONCLUSIONS

This thesis presented Open Information Extraction (Open IE), a powerful new paradigm that eschews relation-specific extraction in favor of process in which relations of interest are automatically discovered from natural language text. Unlike traditional IE systems that require development of training data and corpus analysis with the naming of each new relation, Open IE uses domain-independent methods to learn to identify instances of an unbounded set of relationships in a single data-driven pass.

This thesis also introduced `TEXTRUNNER`, a fully implemented Open IE system that supports a broad range of unanticipated queries over arbitrary relations. `TEXTRUNNER` identifies its own training examples, learns a relation-independent extractor for the English language in the form of a conditional random field, and identifies relation synonyms using an unsupervised algorithm. From 500 million Web pages, `TEXTRUNNER` was found to extract approximately 218 million distinct facts involving both named entities and abstract objects. Of those, 13.5 million describe more than 16,000 relationships about 4.2 million named entities with a precision of 84%. Relative to recent efforts to acquire a diverse set of knowledge from Wikipedia, the set of relationships discovered by `TEXTRUNNER` is larger by an order of magnitude and covers a more diverse set of phenomena.

We found that Open IE, as embodied by `TEXTRUNNER`, achieved precision comparable to relation-specific extractors trained from hundreds, and sometimes thousands, of labeled examples per relation. While the recall of `TEXTRUNNER` was initially lower than traditional, supervised extractors, we developed a self-supervised algorithm that improved the recall of specific relations on demand. Without incurring any additional labor costs, this method nearly tripled the recall of `TEXTRUNNER` and surpassed the recall of a supervised extractor.

We conclude that when relationships in a corpus are not known, or their number is massive, such as on the Web, Open IE is essential. To further improve the power of Open

IE, we have identified several directions for future work.

Extractor-Level Improvements Certain improvements to TEXTRUNNER’s entity identification process have the potential to improve the system’s overall precision and recall. These include integration with a Web-scale named-entity recognizer such as LEX [29], developing the ability to resolve underspecified entities (*e.g.* (*the company, led by, Steve Jobs*) refers to Apple Computer), and finding a way for TEXTRUNNER to disambiguate entity mentions (*e.g.*, does (*Will Smith, arrived in, Florida*) refer to the actor or the athlete?).

Application to Other Languages One particular challenge of interest involves the application of the Open IE paradigm to languages other than English. TEXTRUNNER leveraged an English parser as a resource for automatically providing domain-independent training examples. While parsing technologies exist for several widely-spoken languages, including Chinese, German, French and Spanish, their existence is not guaranteed for all possible target languages. In this case, one must explore other means for acquiring a large set of relation-independent training data.

Textual Theories and Inference While IE systems such as TEXTRUNNER can uncover assertions about individual entities, the formation of a coherent *theory* from a textual corpus involves representation and learning abilities not currently achievable by today’s IE systems. Compared to individual assertions output by IE systems, a theory includes collective knowledge about general concepts. This may include knowledge of classes of entities and their properties, (*e.g.* *Fruit* is something that GROWS), knowledge of relationships at the class level, (*e.g.* CONTAIN(*Fruit, Nutrient*)) and rules that encode dependencies among propositions (*e.g.* GROWNIN(X, Y) & ISA($X, CitrusFruit$) \rightarrow HASWARMCLIMATE(Y)).

The ALICE system [6] demonstrated it could discover high-level concepts and relations among them from the output of TEXTRUNNER. To facilitate the learning of rules that would enable an Open IE system to *reason* over the facts it has acquired, the system must be integrated with inference. For instance, a query asking “*What vegetables help prevent osteoporosis?*” may reveal few sentences stating directly that “*Kale prevents osteoporosis.*”

To answer the question, a system must piece together the answer from multiple sources, such that *kale is a vegetable*, *kale contains calcium* and *doctors believe that calcium prevents osteoporosis*.

While methods for textual inference have typically applied at the sentence or paragraph level, the HOLMES system [73] recently demonstrated it could use assertions output by TEXTRUNNER applied to millions of Web pages to answer complex queries. Further work is needed to extend HOLMES's ability to answer a wide variety of query types and to make it run at interactive speeds. The development of scalable inference methods is a critical step that will help us achieve our original vision of tomorrow's search engine.

BIBLIOGRAPHY

- [1] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Procs. of the Fifth ACM International Conference on Digital Libraries*, 2000.
- [2] S. Auer, C. Bizer, J. Lehmann, G. Kobilarov, R. Cyganiak, and Z. Ives. Dbpedia : A nucleus for a web of open data. In *6th International Semantic Web Conference*, 2007.
- [3] O. Babko-Malaya, M. Palmer, N. Xue, A. Joshi, and S. Kulick. Proposition bank II: Delving deeper. In *HLT/NAACL Workshop on Frontiers in Corpus Annotation*, 2004.
- [4] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 26–33, 2001.
- [5] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *Proceedings of IJCAI*, pages 2670–2676, 2007.
- [6] M. Banko and O. Etzioni. Strategies for lifelong extraction from the web. In *Proceedings on ACM Conference on Knowledge Capture*, 2007.
- [7] M. Banko and O. Etzioni. The tradeoffs between traditional and open relation extraction. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2008.
- [8] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*, pages 92–100, 1998.
- [9] E. Brill. *Lecture Notes in Computer Science*, chapter Processing Natural Language without Natural Language Processing, pages 179–185. Springer Berlin / Heidelberg, 2003.
- [10] Eric Brill and Grace Ngai. vs. machine: A case study in base noun phrase learning. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 65–72, 1999.

- [11] S. Brin. Extracting Patterns and Relations from the World Wide Web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98*, pages 172–183, Valencia, Spain, 1998.
- [12] R. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2007.
- [13] R. Bunescu and R. J. Mooney. Subsequence kernels for relation extraction. In *Proceedings of the 19th Conference on Neural Information Processing Systems (NIPS)*, 2005.
- [14] M. J. Cafarella, D. Downey, S. Soderland, and O. Etzioni. KnowItNow: Fast, scalable information extraction from the web. In *Proceedings of EMLNP*, 2005.
- [15] Michael J. Cafarella, Michele Banko, and Oren Etzioni. Relational web search. Technical Report 06-04-02, University of Washington, 2006.
- [16] Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. Relation extraction using label propagation based semi-supervised learning. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 129–136, Morristown, NJ, USA, 2006. Association for Computational Linguistics.
- [17] Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. Towards the self-annotating web. In *Proceedings of WWW*, pages 462–471, 2004.
- [18] P. Clark, P. Harrison, and J. Thompson. A knowledge-driven approach to text meaning processing. In *Proceedings of the HLT Workshop on Text Meaning Processing*, pages 1–6. ACL Press, 2003.
- [19] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.
- [20] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 1999.
- [21] A. Culotta and J.Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2004.
- [22] A. Culotta and A. McCallum. Confidence estimation for information extraction. In *Proceedings of the Conference on Human Language Technologies / North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2004.

- [23] A. Culotta, A. McCallum, and J. Betz. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proceedings of the Conference on Human Language Technologies / North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 296–303, 2006.
- [24] James R. Curran. Ensemble methods for automatic thesaurus extraction. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 222–229, 2002.
- [25] James R. Curran and Miles Osborne. A very very large corpus doesn’t always yield reliable estimates. In *Proceedings of the 6th conference on Natural language learning*, pages 1–6, 2002.
- [26] V.R. de Sa. Learning classification with unlabeled data. In *Advances in Neural Information Processing Systems*, volume 6, pages 112—119, 1994.
- [27] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, 2004.
- [28] P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24(2):141–168, 1996.
- [29] D. Downey, M. Broadhead, and O. Etzioni. Locating complex named entities in web text. In *Proceedings of IJCAI*, 2007.
- [30] D. Downey, O. Etzioni, and S. Soderland. A probabilistic model of redundancy in information extraction. In *Proceedings of IJCAI*, 2005.
- [31] Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. Web question answering: Is more always better? In *Proceedings of SIGIR*, 2002.
- [32] B. Van Durme and L. K. Schubert. Open knowledge extraction using compositional language processing. In *Symposium on Semantics in Systems for Text Processing (STEP 2008)*, 2008.
- [33] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.
- [34] D. Freitag. Multistrategy learning for information extraction. 1998.
- [35] D. Freitag. Machine learning for information extraction in informal domains. *Machine Learning*, 39(2-3):169–202, 2000.

- [36] D. Freitag and A. McCallum. Information extraction with hmm structures learned by stochastic optimization. In *Proceedings of AAAI*, 2000.
- [37] J. Carroll G. Minnen and D. Pearce. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223, 2001.
- [38] D. Gildea. Corpus variation and parser performance. In *Proceedings of the Conference on Empirical Methods in NLP (EMNLP)*, 2001.
- [39] R. Girju, A. Badulescu, and D. Moldovan. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1), 2006.
- [40] P. Domingos H. Poon. Joint inference in information extraction. In *Proceedings of AAAI*, 2007.
- [41] M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Procs. of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.
- [42] D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of ACM SIGIR*, 1993.
- [43] I. Titov and J. Henderson. Porting statistical parsers with data-defined kernels. In *Proc. Tenth Conf. on Computational Natural Language Learning*, 2006.
- [44] J. Jiang and C. Zhai. A systematic exploration of the feature space for relation extraction. In *Proceedings of the Conference on Human Language Technologies / North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2007.
- [45] Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 209–216, 2006.
- [46] M. Palmer K. Kipper and O. Rambow. Extending Propbank with Verbnet semantic predicates. In *AMTA Workshop on Applied Interlinguas*, 2002.
- [47] Frank Keller, Maria Lapata, and Olga Ourioupina. Using the web to overcome data sparseness. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 230–237, 2002.
- [48] J. Kim and D. Moldovan. Acquisition of semantic patterns for information extraction from corpora. In *Procs. of Ninth IEEE Conference on Artificial Intelligence for Applications*, pages 171–176, 1993.

- [49] D. Klein and C. Manning. Accurate unlexicalized parsing. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2003.
- [50] D. Klein and C. D. Manning. Fast exact inference with a factored model for natural language parsing. In *In Advances in Neural Information Processing Systems 15*, pages 3–10, 2003.
- [51] Cody C. T. Kwok, Oren Etzioni, and Daniel S. Weld. Scaling question answering to the web. In *World Wide Web*, pages 150–161, 2001.
- [52] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Procs. of ICML*, 2001.
- [53] Dekang Lin. Dependency-based evaluation of Minipar. In *In Workshop on the Evaluation of Parsing Systems*, 1998.
- [54] Dekang Lin and Patrick Pantel. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360, 2001.
- [55] A. McCallum. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>, 2002.
- [56] G. A. Miller. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):31–41, 1995.
- [57] Preslav Nakov and Marti Hearst. Using the web as an implicit training set: application to structural ambiguity resolution. In *Proceedings of the Conference on Human Language Technologies / Empirical Methods in NLP (HLT/EMNLP)*, pages 835–842, 2005.
- [58] Grace Ngai and David Yarowsky. Rule writing or annotation: cost-ecient resource usage for base noun phrase chunking. In *Proceedings of ACL*, pages 117–125, 2000.
- [59] J. Nivre. *A Man of Measure*, chapter Two Strategies for Text Parsing, pages 440–448. SKY Journal of Linguistics, 2006.
- [60] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kbler, S. Marinov, and E. Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 3(2):95–135, 2007.
- [61] Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. Towards terascale knowledge acquisition. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 771, 2004.

- [62] M. Pasca, D Lin, J. Bigham, A. Lifchits, and A. Jain. Names and similarities on the web: Fact extraction in the fast lane. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2006.
- [63] Fuchun Peng and Andrew McCallum. Accurate information extraction from research papers using conditional random fields. In *Proceedings of the Conference on Human Language Technologies / North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2004.
- [64] S.P. Ponzetto and M. Strube. Deriving a large scale taxonomy from wikipedia. In *Proceedings of AAAI*, pages 1440–1445, 2007.
- [65] Lance A. Ramshaw and Mitchell P. Marcus. Text chunking using transformation-based learning. In *CoRR cmp-lg/9505040*, pages 82–94, 1995.
- [66] A. Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–175, 1999.
- [67] D. Ravichandran and D. Hovy. Learning Surface Text Patterns for a Question Answering System. In *Procs. of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 41–47, Philadelphia, Pennsylvania, 2002.
- [68] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.
- [69] E. Riloff. Automatically constructing extraction patterns from untagged text. In *Procs. of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049, 1996.
- [70] E. Riloff and R. Jones. Learning Dictionaries for Information Extraction by Multi-level Boot-strapping. In *Procs. of AAAI-99*, pages 1044–1049, 1999.
- [71] B. Rosenfeld and R. Feldman. URES : An unsupervised web relation extraction system. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2006.
- [72] D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In *Procs. of CoNLL*, 2004.
- [73] O. Etzioni S. Schoenmackers and D.S. Weld. Scaling textual inference to the web. In *Proceedings of Empirical Methods in Natural Language Processing*, 2008.
- [74] K. Sagae and J. Tsujii. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics*, 2008.

- [75] L. Schubert and M. Tong. Extracting and evaluating general world knowledge from the Brown corpus. In *Proceedings of the HLT/NAACL Workshop on Text Meaning*, 2003.
- [76] Y. Seginer. Fast unsupervised incremental parsing. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2007.
- [77] S. Sekine. On-demand information extraction. In *Proc. of COLING*, 2006.
- [78] Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Conference on Human Language Technologies / North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2006.
- [79] G. Sigletos, G. Paliouras, C. D. Spyropoulos, and M. Hatzopoulos. Combining information extraction systems using voting and stacked generalization. *Journal of Machine Learning Research*, 6:1751,1782, 2005.
- [80] V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularized approach to semi-supervised learning with multiple views. In *Proceedings of the ICML Workshop on Learning with Multiple Views*, 2005.
- [81] R. Snow, D. Jurafsky, and A. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.
- [82] S. Soderland. *Learning Text Analysis Rules for Domain-Specific Natural Language Processing*. PhD thesis, University of Massachusetts, Amherst, 1997.
- [83] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233-272, 1999.
- [84] S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. CRYSTAL: Inducing a conceptual dictionary. In *Proceedings of IJCAI*, pages 1314-21, 1995.
- [85] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A core of semantic knowledge. In *Proceedings of the 16th International World Wide Web Conference (WWW)*, 2007.
- [86] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A large ontology from Wikipedia and WordNet. *Elsevier Journal of Web Semantics*, 2008.
- [87] R.H. Lathrop T.G. Dietterich and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31-71, 1997.

- [88] K.M. Ting and I. H. Witten. Issues in stacked generalization. *Artificial Intelligence Research*, 10:271–289, 1999.
- [89] E. F. Tjong, K. Sang, and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, 2000.
- [90] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*, pages 173–180, 2003.
- [91] Martin Volk. Exploiting the www as a corpus to resolve pp attachment ambiguities. In *Proceedings of the Corpus Linguistics 2001 Conference*, pages 601–606, 2001.
- [92] Dan Weld, Fei Wu, Eytan Adar, Saleema Amershi, James Fogarty, Raphael Hoffmann, Kayur Patel, and Michael Skinner. Intelligence in wikipedia. In *Proceedings of AAAI*, 2008.
- [93] Wikipedia. Website, 2008. http://en.wikipedia.org/wiki/Wikipedia%3A%3ASize_comparisons.
- [94] D. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–260, 1992.
- [95] F. Wu, R. Hoffmann, and D. S. Weld. Information extraction from Wikipedia: Moving down the long tail. In *Proceedings of KDD*, 2008.
- [96] F. Wu and D. S. Weld. Automatically semantifying wikipedia. In *Proceedings of 16th Conference on Information and Knowledge Management (CIKM)*, 2007.
- [97] A. Yates. *Information Extraction from the Web: Techniques and Applications*. PhD thesis, University of Washington, 2007.
- [98] A. Yates and O. Etzioni. Unsupervised resolution of objects and relations on the web. In *Proceedings of the Conference on Human Language Technologies / North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2007.
- [99] D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *JMLR*, 3:1083–1106, 2003.
- [100] B. Zenko and S. Dzeroski. Stacking with an extended set of meta-level attributes and mlr. In *Proc. of ECML*, 2002.
- [101] Z.H. Zhou and M. Li. Semi-supervised regression with co-training style algorithms. In *IEEE Transactions on Knowledge and Data Engineering*, volume 19, pages 1479–1493, 2007.

- [102] X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, 2005.

Appendix A

TEXTRUNNER DATA

We have made the following data available at <http://www.cs.washington.edu/research/textrunner/banko-thesis-data.tar.gz>

- The complete list of concrete relations found by `TEXTRUNNER` after processing each of Wikipedia and the General-Web corpora.
- The set of tuples found by `TEXTRUNNER` for the ten relations evaluated in Table 4.1, for both the Wikipedia and the General-Web corpora.
- A complete list of concrete relations relative to the entity types listed in Table 4.4 for the following systems
 - `TEXTRUNNER`: Wikipedia and General-Web corpora
 - `DBPEDIA`: Wikipedia corpus
 - `YAGO`: Wikipedia corpus
- The list of seed instances found by `TEXTRUNNER`, used to boot the `R-TEXTRUNNER` system described in Section 5.3 and evaluated in Section 5.3.2.