Open Question Answering

Anthony Fader

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

Doctor of Philosophy

University of Washington

2014

Reading Committee:

Oren Etzioni, Chair

Luke Zettlemoyer

Daniel S. Weld

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Open Question Answering

Anthony Fader

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

For the past fifteen years, search engines like Google have been the dominant way of finding information online. However, search engines break down when presented with complex information needs expressed as natural language questions. Further, as more people access the web from mobile devices with limited input/output capabilities, the need for software that can interpret and answer questions becomes more pressing. This dissertation studies the design of Open Question Answering (Open QA) systems that answer questions by reasoning over large, open-domain knowledge bases.

Open QA systems are faced with two challenges. The first challenge is knowledge acquisition: How does the system acquire and represent the knowledge needed to answer questions? I describe a simple and scalable information extraction technique that automatically constructs an open-domain knowledge base from web text. The second challenge that Open QA systems face is question interpretation: How does the system robustly map questions to queries over its knowledge? I describe algorithms that learn to interpret questions by leveraging massive amounts of data from community QA sites like WikiAnswers. This dissertation shows that combining information extraction with community-QA data can enable Open QA at a much larger scale than what was previously possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I'd first like to thank Oren Etzioni for advising me for the past six years. Oren taught me many things, but the most important lesson I learned from him is to focus on finding the right problem to work on. He always challenged me to seek out ambitious projects and to avoid research ratholes and academic cul-de-sacs at all costs. In addition to teaching me deep, fundamental research principles, Oren also made me laugh a lot. I consider him to be the Jerry Seinfeld of computer science.

I'd also like to thank Luke Zettlemoyer for being my unofficial second advisor. I have learned more about natural language processing and machine learning from Luke than from anyone else. He was always there to help me and he has created an amazingly innovative and stimulating research group at UW. Luke is also one of the nicest, most genuine human beings that I know of—he ranks right up there with Mr. Rogers.

Dan Weld has always provided excellent feedback on my work and I'd like to thank him for asking hard-hitting questions. Dan is like my academic godfather and I've been lucky to have him on my quals, generals, and thesis committees (the Dan Weld hat-trick).

I was fortunate enough to work with Doug Downey, who is not only a brilliant researcher, but also holds the record for driest sense of humor east of the Mississippi. I'd also like to thank Stephen Soderland for advising me when I first arrived at UW. Stephen's island lifestyle is an inspiration to many city folk. Speaking of island-dwellers, I'd like to thank Michael Schmitz for introducing me to Scala, wild blueberries, and Thanh Vi.

One of the best parts of grad school has been meeting so many great people. I'd like to give copious shout-outs to my labmates Janara Christensen, Morgan Dixon, and Abe Friesen. They've been my close friends for the last six years and have always been there for me . . . to annoy. Thanks to Brandon Lucia for some of the funniest and deepest conversations I have ever had. I also have many great memories of dinners, barbecues, and hikes with

Nicki Dell, Nell O'Rourke, Franzi Roesner, Pete Hornyak, Todd Schiller, Cliff Johnson, and Greg Akselrod. Thanks to Mike Toomim, Travis Kriplean, and Kevin Miniter for admitting me to the Invisible College of Hawaii.

I'd like to thank Meera Nilekani and Curt Fischer for being hilarious friends and bed-fellows. Also thanks to Allison Reibel for taking me on a life-changing trip around Capitol Hill. Thanks to Jeremy Royal, Jordan Vogt-Roberts, Jon Wilcox, Erin Milbeck Wilcox, and Matt West for being lifelong friends. I'd also like to give a sprout-out to Chances With Wolves, who provided the soundtrack to my grad school experience.

Finally, I'd like to thank my family. Thanks to my sister Jane for being my best friend in Seattle, to my sister Paula for being a spiritual role model, to my dad Hugh for raising me to work hard, and to my mom Amy for raising me to be compassionate.

DEDICATION

to my family and friends from Michigan

Chapter 1

INTRODUCTION

The growth of the web has revolutionized the way people access information. For the past fifteen years, search engines like Google have been the dominant way of finding information online. Search engines allow people to quickly filter the entire web to just small collection of pages that are most relevant to their needs. Research in the field of information retrieval and billions of interactions with real users have resulted in search engines that are fast, accurate, and can serve information about any imaginable topic.

However, the search engine model for information access breaks down for more complex information needs. For example, consider the following question:

Q1: What is illegal in the US but legal in Mexico?

Figure 1.1 shows the output of the Google and Bing search engines for this question. Neither search engine returns a page containing an answer, instead returning pages that are only indirectly related to the question.

Why are Google and Bing unable to answer Q1? First, search engines perform keyword search, and are not designed to handle full, natural language questions. Second, search engines assume that the answer to a question will be explicitly stated on a single web page. For common questions like "What is the capital of France?" this assumption generally holds—the web is large enough that the most frequently asked questions are explicitly answered in text. However, for rare or previously unseen questions, there will be no single page that directly provides an answer. Systems like Wolfram|Alpha can handle some complex information needs, but are often brittle and lack knowledge (as shown in the bottom of Figure 1.1). As it stands, the only way to answer a question like Q1 is to issue multiple queries to the search engine (e.g., "legal in the US", "illegal in Mexico", or even "against the law in Mexico"), skim the resulting pages, and manually combine evidence to obtain an answer. Today's search engines are fundamentally unable to answer these types of questions.

Figure 1.1: Search engine results for the question "What is illegal in the US but legal in Mexico?" using Google (top), Bing (middle), and Wolfram|Alpha. Retrieved on May 1, 2014.

The focus of this dissertation is the design of Question Answering (QA) systems that can directly answer questions like Q1. Instead of mapping keyword searches to a list of documents, QA systems use knowledge to map natural language questions to a list of direct answers. For example, a QA system might map Q1 to the answer "Cuban cigars" by interpreting the question as

"Find a concept x such that: x is illegal in the US $\wedge$ x is legal in Mexico,"

and then searching its knowledge to find that $x =$ "Cuban cigars."

Question Answering offers two main advantages over web search engines. First, QA is declarative: a person states *what* their information need is without specifying *how* the answer should be found. Second, QA is a natural user interface: a person states their information need *in their own words,* which are then interpreted by the QA system. These two properties allow QA systems to satisfy more complex information needs with less human effort than today's search engines. Providing natural, declarative access to information will become even more important as people shift from desktop computers to mobile phones, where entering complex information needs is cumbersome.

1.1 Challenges

Question Answering has been studied for at least fifty years, with the earliest research beginning in the 1960s (Simmons, 1965, 1970). Researchers have demonstrated that with enough manual effort, it is possible to engineer a QA system that can answer questions like Q1 for a particular topic. However, there is no single system that can answer complex questions across many domains. The difficulty in creating such a QA system can be traced to two problems:

1. **Knowledge Acquisition:** How does a QA system acquire and represent the knowledge needed to answer questions? Knowledge acquisition involves encoding assertions like "Cuban cigars are legal in Mexico" or "Cuban cigars are illegal in the US" in a format that can be queried during QA.

	Answer Retrieval	**Semantic Parsing**
Question	Who is Tom Cruise married to?	What borders the state with the highest population?
Knowledge	`...him. Tom Cruise is married to Katie Holmes as of the 18th of November. They have 1 child and ...`	`state(az), state(ca), state(or), borders(az, ca), borders(or, ca), population(ca, 38m), population(az, 7m), ...`
Query	`Tom Cruise is married to ([A-Z].* [A-Z].*)`	$\lambda x.$ `borders(`x`, argmax(state, population))`
Answer	`Katie Holmes`	`az, or`

Table 1.1: A comparison of the two dominant approaches in question answering. Answer retrieval systems interpret questions as patterns that operate directly over text. Semantic parsing systems map questions to a formal meaning representation (*e.g.*, lambda calculus) and operate over a knowledge base.

2. **Question Interpretation:** How does a QA system map questions to queries over its knowledge? Question interpretation involves inferring the information need of a question and then formulating a plan to obtain an answer.

There is a tension between knowledge acquisition and question interpretation. QA systems that can acquire large amounts of knowledge tend to have limited question-interpretation capabilities. On the other hand, QA systems that can interpret complex questions only work over a single domain, and are not general-purpose tools for information access.

For example, answer retrieval systems (Voorhees and Tice, 2000; Prager, 2006) use English text to represent knowledge. The left half of Table 1.1 shows an example of how an answer retrieval system would answer a question. Representing knowledge as text enables easy, domain-scalable knowledge acquisition: simply download as much text as possible from the web. However, answer retrieval systems have limited question-interpretation capabilities and can only handle questions whose answers are directly stated in a single sentence. In other words, answer retrieval has the same limitation as search engines and cannot meet complex information needs like Q1.

At the other end of the spectrum are semantic parsing systems (Grosz et al., 1987; Zelle and Mooney, 1996). The right half of Table 1.1 shows an example of how a semantic parser would answer a question about US geography. Semantic parsing systems use a formal meaning representation language to represent their knowledge. During question answering, the input question is mapped to a query expressed in the same formal meaning representation as the knowledge. This formalism is expressive enough to answer questions that involve reasoning over multiple pieces of information. For example, the query in Table 1.1 computes the largest state by population and then computes its neighboring states. However, a major disadvantage of semantic parsing systems is that encoding knowledge into a formal meaning representation is a non-trivial task that generally requires expert annotators. This limits semantic parsing systems to single domains where the cost of manually encoding knowledge is not too high. Thus, semantic parsing systems excel at question interpretation, but lack the knowledge needed for a general-purpose QA system.

1.2 Overview of Approach

This dissertation discusses work towards *Open QA:* designing a system that is both domain-scalable and able to handle complex information needs. I use the term Open QA to highlight the differences with previous work on answer retrieval and semantic parsing systems. The goal of Open QA is to combine the domain scalability of answer retrieval with the question-interpretation abilities of semantic parsing. Constructing a system that has both of these properties involves scaling knowledge acquisition and question interpretation beyond what has been done in previous work.

My approach to Open QA is based on two ideas. The first idea comes from the observation that there is some information that does not exist in any knowledge base but is expressed in text, *e.g.*, rapidly growing scientific literature, online product reviews, or imprecise assertions like "Bananas are high in potassium." I will use large-scale information extraction techniques to automatically construct an open-domain knowledge base containing this type of information and access it alongside existing knowledge bases. For example, an IE system may transform a sentence like "Cuban cigars, which are illegal in the US..." into a tuple like `(Cuban cigars, illegal in, US)` that represents the assertion that the

`illegal in` relationship holds between the concepts `Cuban cigars` and `US`. The transformation from text to tuples provides a useful abstraction: the information in text can now be queried in terms of concepts and relationships. The information need of a question can then be represented in the same way. For example, the information need of Q1 could be written as `(?x, illegal in, US)` $\wedge$ `(?x, legal in, Mexico)`, where `?x` is an answer variable and each triple encodes a constraint on `?x`.[1] To answer the question, the system can then find all values of $?x$ that satisfy both constraints. Thus, information extraction is a technique for rapidly acquiring assertions that may not be found in any existing knowledge base.

The second idea I use for Open QA is based on the observation that community QA (CQA) sites can provide a learning signal for open-domain question interpretation. Sites like WikiAnswers[2] allow users to ask and answer questions online. For example, Figure 1.2 shows the WikiAnswers page for the question "Are sugar gliders legal in the US?" Below the question, another user has posted an answer and cites references. In addition to posting questions and answers, users also organize questions by merging equivalent questions into a single page. Figure 1.3 shows the alternate wordings of the question from Figure 1.2, *e.g.* "Is it legal to own a sugar glider in the US?" CQA sites are maintained by millions of users posting questions and answers about almost any topic. This data is a valuable resource for constructing an Open QA system: it provides millions of examples of how people ask, answer, and reword questions across many different domains.

My approach can be summarized in the following thesis statement:

> Open QA can be enabled by (1) acquiring open-domain knowledge via large-scale information extraction, and (2) learning to interpret questions from naturally occurring data on community QA sites.

To support this statement, I will:

- Present a novel information extraction technique to automatically construct a massive knowledge base from web text.

[1] This query language is discussed further in Chapter 5.

[2] `http://wiki.answers.com`

Answers.com > Wiki Answers > Categories > Animal Life > Mammals > Land Mammals > Marsupials > Sugar Gliders > Are sugar gliders legal in the US?

Are sugar gliders legal in the US?

In: Sugar Gliders [Edit categories]

Answer:

Sugar gliders are legal in most states, while other states require a permit.

Some states where sugar gliders are absolutely illegal are:

Alaska (refer to: Alaska Department of Fish and Game)

California (refer to: California Department of Fish and Game)

Hawaii

Some states which require a permit:

Massachusetts

Maine

Pennsylvania Although it states that they will no longer issue permits, basically making them illegal. http://www.pgc.state.pa.us/

Utah

They are legal in:

Alabama

Arkansas

Arizona (According to Bill Van Pelt, Nongame Bird and Mammal Program Manager of the Arizona Game and Fish Department)

Connecticut

Florida (but Florida does have minimum cage size requirements)

Figure 1.2: WikiAnswers page for the question "Are sugar gliders legal in the US?" Downloaded on May 2, 2014.

Contributor asked **Is having a sugar glider legal in us** and said it was the same as **Are sugar gliders legal in the US** 3 May 2011 20:15

Contributor asked **Are sugar gliders legal in maryland** and said it was the same as **Are sugar gliders legal in the US** 20 Feb 2010 18:16

Contributor asked **Are shugar gliders legal in the us** and said it was the same as **Are sugar gliders legal in the US** 17 Feb 2010 17:18

Contributor asked **Are sugar gliders legal in US** and said it was the same as **Are sugar gliders legal in the US** 16 Feb 2010 03:38

Contributor asked **How many states in the US is it legal to own a Sugar Glider** and said it was the same as **Are sugar gliders legal in the US** 21 Jan 2010 04:34

Contributor asked **Are sugar gliders legal in freeport** and said it was the same as **Are sugar gliders legal in the US** 15 Jan 2010 02:30

Ashcash10 [2] asked **Is it legal to own a sugar glider in the US** and said it was the same as **Are sugar gliders legal in the US** 12 Jan 2010 20:20

Contributor asked **Can you own sugar gliders in the US** and said it was the same as **Are sugar gliders legal in the US** 9 Dec 2009 22:13

Figure 1.3: Alternate wordings of the WikiAnswers question "Are sugar gliders legal in the US?" Downloaded on May 2, 2014.

- Use the data from CQA sites to learn a question-interpretation function that maps questions to queries over the extracted knowledge base.

1.3 Summary of Contributions

I will present three main contributions in support of my thesis statement:

1. **Identifying Relations for Open Information Extraction (Chapter 3)**, which focuses on acquiring open-domain knowledge using a novel information extraction technique. This work was first published in 2011 at EMNLP (Fader et al., 2011).

2. **Paraphrase-Driven Learning for Open QA (Chapter 4)**, which focuses on robust question interpretation using the paraphrase information available on WikiAnswers. This work was first published in 2013 at ACL (Fader et al., 2013).

3. **Open QA Over Curated and Extracted Knowledge Bases (Chapter 5)**, which focuses on combining knowledge from multiple sources and improving the accuracy of question interpretation. This work was first published in 2014 at KDD (Fader et al., 2014).

In the following three sections, I will provide a high-level summary of each contribution.

1.3.1 Identifying Relations for Open Information Extraction

Open Information Extraction (Open IE) is a technique for identifying relationships in text. Open IE systems take sentences as input and output `(argument1, relation phrase, argument2)` triples. For example, consider the following sentence:

> "Windsor also does business in Cuban cigars, which are banned in the US."

An Open IE system might extract two triples from this sentence: `(Windsor, does business in, Cuban cigars)` and `(Cuban cigars, banned in, US)`. A major challenge in Open IE is identifying the relation phrases in the sentence. For example, an Open IE system must identify that `are banned in` is a relation phrase, but `which are` is not.

I will describe a system called ReVerb that implements a fast, simple, and domain-independent technique for identifying relation phrases in English text. Previous approaches identify relation phrases using machine-learned models that are trained on heuristically labeled examples. These approaches generate false-positive predictions, which flood their output with incoherent relation phrases like `which are` and underspecified relation phrases like `does`. Further, existing approaches systematically miss relation phrases expressed via light verb constructions like `made a deal with`. Instead of learning a model of relation phrases from heuristically labeled data, ReVerb uses a compact pattern expressed over part-of-speech tags to identify relation phrases. I demonstrate that this pattern covers 85% of relation phrases expressed with verbs, and show that it outperforms existing approaches in terms of precision and recall.

ReVerb has been run on web-scale datasets to produce a large extracted knowledge base, a sample of which is shown in Table 1.2. The output from ReVerb is used as knowledge for the QA systems discussed in the next sections.

1.3.2 Paraphrase-Driven Learning for Open QA

One of the desired properties of a QA system is to be robust to the variations in natural language questions in order to provide a natural, declarative interface. For example, a QA system should be able to infer that all of the questions in Table 1.3 are different ways of asking "Is online gambling legal in the US?" and route them to the same answer. I describe a system called Paralex that learns a robust question-interpretation function from the paraphrase data available on WikiAnswers.

The Paralex system makes two technical contributions. First, Paralex uses ReVerb as a source of knowledge and is the first system to perform Open QA over an extracted knowledge base. Second, Paralex uses a novel learning algorithm that generalizes from millions of paraphrase clusters like the example in Table 1.3. For example, Paralex aligns these questions and infers paraphrase information like:

- "america" can be paraphrased to "the US"
- "betting online" can be paraphrased to "online casinos"

Argument 1	Relation Phrase	Argument 2
Gambling	is banned in	Islam
Foie gras	is not banned in	California
cocaine use	was legal in	the United States
Large breeds	were banned in	Beijing
Prostitution	is legal in	Amsterdam
Independent unions	are illegal in	China
Mah Jong	was completely banned in	China
Pyramid selling	is illegal in	Australia
GTA4	is banned in	the UAE.
humor	is illegal in	Poland
Hitch hiking	is illegal in	Oz
Dog and cat fur	should be banned in	Europe
same-sex marriages	were legalized in	California
common law marriages	are declared illegal in	England
Homosexuality	is illegal in	Mauritius
Cock fighting	was banned in	1849
the WOW	has been banned in	Manchester
Sabots	are illegal in	Colorado
gay marriage	became legal in	Massachusetts
poker	was recently legalized in	Catalonia
Slavery	is declared illegal in	the Oregon Country
Pornography	is legal in	Australia
Corporal punishment	is legal in	Wilkinson County

Table 1.2: A sample of ReVerb extractions containing the strings "illegal," "legal," or "banned" in the relation-phrase field.

Can us citizens gamble online?
Can you gamble online in america?
Internet gambling is legal in the US?
Is betting online legal in US?
Is it illegal to do online gamble?
Is it legal to gamble online in the US?
Is it legal to gamble online in america?
Is online casinos legal in the us?
Is online gambling forbidden in the US?
Online gambling in US is legal?

Table 1.3: An example cluster of questions that users on WikiAnswers have tagged as being paraphrases.

- "Can you X in Y?" can be paraphrased to "Is X legal in Y?"

PARALEX uses this generalized paraphrase information to interpret new questions that are unseen in the training corpus. I provide experimental results demonstrating that PARALEX achieves higher precision and recall than a hand-constructed QA system that does not learn from paraphrases. I also release the WikiAnswers paraphrase corpus for public use, which contains over 20 million question clusters and includes inferred word alignments between questions.

1.3.3 Open QA Over Curated and Extracted Knowledge Bases

PARALEX is the first system to perform Open QA over an extracted knowledge base, but has limitations. First, while PARALEX was able to achieve high accuracy on questions that are known to be answerable using the REVERB knowledge base, PARALEX's accuracy drops sharply when presented with unfiltered questions. Second, PARALEX is unable to leverage knowledge from other sources, including other extracted knowledge bases as well as curated knowledge bases like Freebase (Bollacker et al., 2008). Finally, PARALEX is limited to

simple questions and is unable to answer questions that involve joining multiple pieces of information together.

My final contribution is an Open QA system called OQA that overcomes the problems of PARALEX. OQA decomposes the full question-answering problem into smaller problems that are easier to solve. Figure 1.4 shows an example of how OQA maps an input question to an answer by chaining together a sequence of operators. These operators include paraphrasing the input question, parsing the question into a query, and rewriting the query to match the knowledge base. OQA uses a lightweight query language that allows it to access multiple knowledge bases through a single interface. Finally, OQA learns a robust scoring model that allows it to measure the confidence of its answers. I demonstrate that OQA achieves higher precision and recall than PARALEX on three external question sets. I also demonstrate that OQA benefits from being able to access multiple sources of knowledge, and quantify the benefits of each knowledge source.

1.4 Outline

The rest of this dissertation is organized as follows. Chapter 2 provides information about question answering and scopes the problems addressed in this dissertation. Chapters 3, 4, and 5 will describe the REVERB, PARALEX, and OQA systems in detail. Finally, 6 will conclude with a summary and discussion of future work.

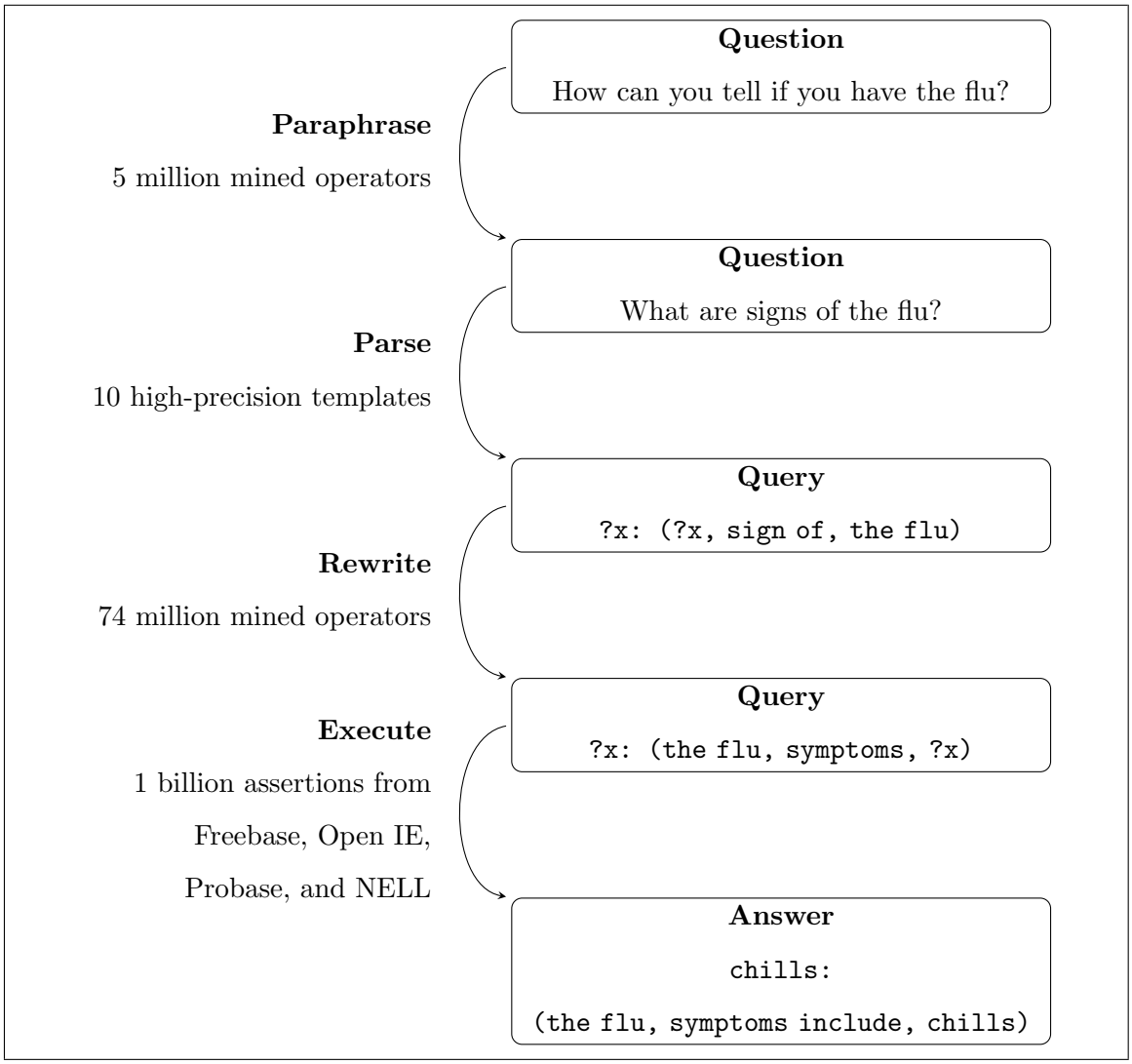


Figure 1.4: An example of how the OQA system maps the question "How can you tell if you have the flu?" to the answer "the chills."

Chapter 2
BACKGROUND AND RELATED WORK

Question answering has been studied extensively and there is a large body of literature. I will not attempt to give a comprehensive survey of the entire field. Instead, I will focus on the different approaches to knowledge acquisition and question interpretation and examine how they relate to Open QA. For a more complete history of QA, I recommend the following survey papers:

- Simmons' survey papers give an account of the first generation of QA systems from the 1960s (Simmons, 1965, 1970).

- Hirschman and Gaizauskas provide an history of QA up to 2001 (Hirschman and Gaizauskas, 2001).

- Prager also provides a history of QA until 2007 and provides a comprehensive summary of open-domain answer retrieval (Prager, 2006).

In Section 2.1, I will define the scope of the rest of the dissertation and highlight other formulations of the question answering problem. Then, I will describe previous work on knowledge acquisition (Section 2.2) and question interpretation (Section 2.3).

2.1 Problem Scope

The name "question answering" is ambiguous and depends on how "question" and "answer" are defined. The broadest definition of question answering could define a question as any information need expressed in natural language and an answer as anything that meets the information need. Instead of attempting to solve this very general problem, I will focus on the sub-problem of handling factoid questions, which can be answered with a set of short, string answers.

	Question	Answer
1	What is the capital of France?	Paris
2	Name some primates that live in the jungle.	Gorillas, spider monkeys, howler monkeys
3	How does a fish breathe?	Gills
4	What is potassium?	chemical element, nutrient, alkali metal
5	Restaurants in Seattle	Hot Mama's Pizza, Fogon, Rancho Bravo

Table 2.1: Example factoid questions and answers.

The input to a factoid QA system is a single question and the output is a list of string answers. Table 2.1 lists some example factoid questions and answers. The answer to a factoid question may have a single answer (as in rows 1 and 3) or many answers (as in rows 2, 4, and 5). I do not require that factoid questions meet any syntactic criteria. For example, factoid questions do not necessarily start with a Wh-word (as in row 3). Further, factoid questions are not even necessarily linguistically valid questions, and can be expressed as a command (like row 2) or even as an incomplete sentence (like row 5).

My definition of factoid question is intentionally vague and includes questions that are not considered factoid questions by other researchers.[1] Defining a taxonomy of questions or information needs is itself a challenging research problem (Graesser and Person, 1994; Broder, 2002; Nielsen et al., 2008). For example, Voorhees and Tice carefully and precisely document their criteria for constructing a set of factoid questions to evaluate the

[1]The answer retrieval community has its own definition of factoid questions (Voorhees and Tice, 2000; Prager, 2006) that does not include definition questions (like row 4) or questions with more than one answer (like rows 2, 4 and 5).

performance of answer retrieval systems (Voorhees and Tice, 2000). In this dissertation, I deliberately avoid going through this process and instead use either existing evaluation sets or naturally occurring questions from community QA sites for evaluation.

There are different formulations of QA that fall outside the scope of factoid questions. I include brief summaries of these alternative problem formulations to convey related technical problems that I do not attempt to solve. Other problem formulations include:

- **Context-Dependent QA:** The input/output interface of factoid QA treats each input question independent of the others. An alternative problem formulation is to answer questions in the context of a particular conversation or world state. Tracking a dynamic context is often essential for handling information needs that go beyond simple factoid questions. A common case where this extra functionality is needed includes spoken dialogue systems. For example. modeling a conversation with an automated airline-booking system (Dahl et al., 1994; Miller et al., 1996; Zettlemoyer and Collins, 2009; Artzi and Zettlemoyer, 2013) is a common testbed for this type of question answering. Another common scenario is instructing a computer to carry out a sequence of actions interactive environments (Winograd, 1971; Roy et al., 2000; Matuszek et al., 2012; Branavan, 2012). Understanding context-dependent language is a more general problem than the factoid QA problem and scaling context-dependent QA systems to an open-domain setting is an area of ongoing research (Hixon and Passonneau, 2013).

- **Reading Comprehension:** Another alternative formulation for QA involves answering a question using a short passage of background text (Charniak, 1972; Lehnert, 1977; Hirschman et al., 1999; Jansen et al., 2014). A common example of this is reading comprehension, where a system processes a short story and then answers multiple-choice questions about the text. This problem formulation is different from factoid QA in two ways. First, the questions in reading comprehension are focused on the given background text, whereas in factoid questions are answerable using general world knowledge. Second, the questions that appear in reading comprehension tend

to be more challenging and require understanding the discourse structure of the text. For example, consider this text from the MCTest corpus (Richardson et al., 2013):

> Then [James] walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home.

A reading comprehension system would be responsible for answering the question "What did James do after he ordered the fries?" Answering this question requires solving problems like coreference resolution (Ng, 2010), identifying temporal relations (Verhagen et al., 2007), and recognizing textual entailment (Dagan et al., 2006). This level of text understanding goes beyond the techniques used in this dissertation.

- **Automatic Summarization:** All of the QA formalisms described above involve returning a set of answers or actions as output. However, for open-ended questions like "What is the Edward Snowden scandal all about?" a single answer is not sufficient. For questions like these, the answer needs to be a high-level description summarizing the target concept. Multi-document summarization (MDS) systems take a collection of sentences (Gupta and Lehal, 2010; Das and Martins, 2007) with an optional query (Carbonell and Goldstein, 1998) as input and return a subset of sentences that summarize the collection. Recent work has started to move MDS beyond just selecting sentences and towards providing more structure (Christensen et al., 2013). These summarization systems can be thought of as QA with a much more complex output space than an answer set.

In the future, a single QA system may be able to handle all of the QA challenges described above. For now, the approaches to these problems are all different enough that they are largely disjoint areas of research.

2.2 Knowledge Acquisition

In this section, I give an overview of the different approaches to constructing large, open-domain knowledge bases (KBs) that go beyond the textual representation used by answer

retrieval systems. I will focus on open-domain knowledge acquisition techniques (as opposed to single-domain techniques) and their applications to QA.

There are two main approaches to constructing open-domain knowledge bases: curation and extraction. Curated KBs are constructed by people who manually enter rules and assertions into a formal representation language. Extracted KBs are constructed by information extraction systems that convert text into structured assertions. The principal trade-off between these approaches is domain-scalability versus accuracy. Curated KBs have precise, comprehensive coverage of some domains, but may lack information about others. In contrast, extracted KBs generally have broad coverage across all domains, but are often incomplete within any particular domain and may contain extraction errors.

Beyond these qualitative characterizations, there has been little work on understanding how different KBs affect QA system performance. Most research compares different approaches to question interpretation and evaluates them on questions that are filtered to be answerable using a particular KB. There have been no controlled experiments that test the effect of different KBs on system performance.[2] Because of the lack of direct comparisons, in the next two sections I will list what KBs are available and to what extend they have been used in QA.

2.2.1 Curated Knowledge Bases

One of the most famous curated KBs is Cyc (Lenat et al., 1990), an ongoing project where experts encode open-domain, commonsense knowledge using a formal knowledge-representation language. Cyc has been used in QA system in various ways. For example, Cyc has been used as a sanity-checker for answer retrieval (Chu-Carroll et al., 2004) and for selecting answers in multiple-choice tests (Friedland et al., 2004). The only QA system that uses Cyc to generate candidate answers is MySentient (Curtis et al., 2005). However, MySentient was not evaluated on a question set, so its utility for Open QA is unclear.

In the START system (Katz, 1997), experts annotate text and other media with machine-interpretable annotations, which can then be used for question answering. START was

[2] I address this experiment with the OQA system in Chapter 5.

shown to answer 67% of the questions presented by its users (Katz et al., 2004), but there has been no quantitative understanding of the coverage of START relative to other knowledge sources.

While Cyc and START are primarily curated by experts, other projects have taken a crowdsourcing approach, where knowledge is curated by an online community. For example, DBpedia (Auer et al., 2007) is the center of the Linked Open Data initiative (Bizer et al., 2008) and compiles the information from Wikipedia infoboxes into a single source. YAGO (Suchanek et al., 2007) combines Wikipedia infoboxes with WordNet (Fellbaum, 1999). WikiNet (Nastase et al., 2010) generates a knowledge base from the information on Wikipedia category and infobox annotations. Freebase (Bollacker et al., 2008) provides an interactive authoring tool that allows users on the web to collaboratively edit an ontology and add assertions to an open-domain KB. Like Cyc, these KBs have been used to augment answer retrieval in various ways. For example, YAGO has been used for answer typing (Kalyanpur et al., 2011) and Freebase was used for "answer lookup" in the IBM Watson system (Chu-Carroll et al., 2012).

Recently, researchers have started building QA systems that use these curated KBs as their primary source of knowledge. Questions have been interpreted as formal queries over curated KBs like DBpedia (Unger et al., 2012; Walter et al., 2012), YAGO (Moussa and Abdel-Kader, 2011), and Freebase (Cai and Yates, 2013; Berant et al., 2013; Kwiatkowski et al., 2013). This new thread of research has led to the construction of benchmark question sets. The Question Answering Over Linked Data (QALD) competitions[3] aim to compare QA systems on a set of test questions answerable over DBpedia. Several benchmark question sets have been created for Freebase, including the Free917 question set (Cai and Yates, 2013) and the WebQuestions question set (Berant et al., 2013). However, there has yet to be a comparison of these different KBs on the same question set.

[3] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/`

2.2.2 Extracted Knowledge Bases

The incompleteness of curated KBs (Pratt, 1994) led researchers to consider extracting open-domain knowledge from text. The first major project to attempt this was the Open Mind project at MIT (Stork, 1999), where non-expert users enter commonsense knowledge expressed in English sentences. These sentences are then converted to formal KB assertions using hand-written extractor patterns. Open Mind was the first system to demonstrate the utility of extracting open-domain knowledge from text. However, the Open Mind KB was never used in an end-to-end Open QA system.

After Open Mind, researchers began using information extraction to construct knowledge bases from large text corpora. There have been two main approaches to constructing a KB using information extraction. The first approach is automatic KB completion (AKBC), where the system assumes access to a set of target relationships (*e.g.*, a schema or ontology) and a set of example instances of the target relationships. AKBC can be thought of as a hybrid of curation and extraction approaches: people curate the KB schema, and IE systems extract instances from text to populate the KB. AKBC techniques have been used to extend Freebase (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011) and YAGO (Suchanek et al., 2009; Kasneci et al., 2009). The NELL system (Carlson et al., 2010) uses its own schema and example instances, but does not contain the broad coverage of the previous three KBs. There has been no research studying the benefits of AKBC on the end-to-end performance of a QA system, for example comparing a QA system's performance using an initial KB to the performance after performing AKBC.

The second way to construct an extracted KB is to have the IE system itself identify and extract relations from text. Under this paradigm, the system makes no commitment to a particular ontology *a priori* and instead lets the text itself define the relations and concepts. This approach has been referred to in multiple ways, including Preemptive IE (Shinyama and Sekine, 2006), On-Demand IE (Sekine, 2006), Open Knowledge Extraction (Van Durme and Schubert, 2008), and Open IE (Banko et al., 2007). Open IE was the first of these systems to be designed for web-scale IE, where speed and robustness to unconstrained text are necessary. Chapter 3 provides a more in-depth, technical description of the different

approaches to Open IE.

There has been little work applying the output of large-scale, open-domain IE to question answering. Open IE has been used as a way to align the natural language in questions with concepts in Freebase (Cai and Yates, 2013; Berant et al., 2013), but outside of the work described in this dissertation, there has been no Open QA system built on top of an extracted knowledge base.

2.2.3 Summary of Knowledge Acquisition

Research over the past twenty years has resulted in an explosion of open-domain knowledge bases, both curated and extracted. At this time, there is not a good understanding of how the various KBs perform when used for Open QA. Chapter 5 describes the first work to explore the trade-offs between these different KBs in a quantitative manner.

2.3 Question Interpretation

Once a source of knowledge has been acquired, a QA system needs to solve the problem of question interpretation: mapping a person's input question to an executable query over the knowledge. In this section I describe how previous systems have solved this problem and to what extent these approaches can be used for Open QA.

A key challenge to question interpretation is the vocabulary mismatch problem. For example, consider the question "How can I tell if I have the flu?" The answer to this question may be expressed in text as "Fever is a symptom of the flu" or in a knowledge base as `(flu, symptoms, fever)`. A question interpreter needs to bridge the gap between the way the information need is expressed in the question and the way the answer is expressed in the knowledge. In this example, the system must infer that the question is asking about the *symptoms* of the flu, despite the string never being used in the question itself. Solving this problem is necessary for building a QA system that is natural and declarative, as described in Chapter 1.

The way a QA system performs question interpretation depends largely on the format and scale of its knowledge. Answer retrieval systems generally break the question interpretation problem into two steps: a keyword search over the corpus, and then an answer-

extraction step (see Prager's survey for a full description). The answer extraction step can be done using keyword proximity, applying learned question-answer templates (Ravichandran and Hovy, 2002), or aligning dependency parse patterns between the question and text (Punyakanok et al., 2004). These approaches are not directly applicable to Open QA, where queries are expressed over a structured knowledge base, not text.

Question interpretation has been the main focus of research in semantic parsing. Early research demonstrated that with enough hand-engineering, it is possible to construct a QA system that works well on a single domain (Woods, 1973). Research then shifted towards *portable* QA systems that minimize the amount of human effort that goes into porting the system to a new domain. The authors of the TEAM system (Grosz et al., 1987) achieved portability by carefully factoring the system architecture into reusable, domain-independent components and domain-specific components. In the 1990s, researchers began building QA systems that learn the domain-specific components from data, which further improved portability (Miller et al., 1994). A major research direction became reducing the amount of supervision and expertise needed to train the domain-specific components (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Clarke et al., 2010; Liang et al., 2011).

Machine-learning techniques have greatly reduced the burden needed to port semantic parsers to new domains. However, the amount of training data required by these techniques is directly tied to the number of concepts and relations in the knowledge base. This is because these systems use the training data to extract a mapping from natural language to KB concepts. The learners are unable to generalize to KB concepts that are unseen in the training data. Thus, to scale these approaches to large, open-domain KBs, the systems either need access to a much larger training set, greater generalization from a smaller training set, or both.

Recently there has been some work attempting to scale semantic parsers to large, open-domain KBs. Several systems bootstrap a mapping from natural language to Freebase concepts by aligning Open IE tuples with Freebase (Cai and Yates, 2013; Berant et al., 2013). Kwiatkowski et al. factor the question interpretation process into a two-step process, where first a KB-independent parse of the input question is identified and then performs on-the-fly ontology matching, guided by a smaller amount of training data (Kwiatkowski

et al., 2013). It has yet to be shown whether these techniques can be applied to the larger, noisier extracted knowledge bases.

The Open QA systems described in this dissertation build upon previous work in portability by leveraging from more data and aiming for greater generalization from training data. The PARALEX system demonstrates how community QA sites can be leveraged to train a question-interpretation function at a much larger scale than what has been done in previous work. The OQA system shows how using a low-dimensional, unlexicalized feature representation for question interpretation can lead to better results than existing systems. OQA is also the first system to demonstrate that combining curated and extracted KBs can result in better performance, while the work in semantic parsing has only been demonstrated to work with curated knowledge bases.

2.4 Summary of Related Work

Most previous work in open-domain QA has been limited to answer-retrieval systems, which have limited question-interpretation abilities, or semantic-parsing systems, which have limited knowledge. The rise of large-scale, open-domain knowledge bases, both curated and extracted, provide a new opportunity to combine the domain-scalability of answer retrieval with the question interpretation of semantic parsing. However, previous question-interpretation techniques cannot be directly applied to these large KBs. The work described in this dissertation learns to interpret questions using data from community-authored QA sites and enables Open QA at a much larger scale than what was previously possible.

The next chapters present the technical contributions. In Chapter 3, I will describe the REVERB Open IE system, which I use to extract a large, open-domain KB from web text. Then in Chapter 4, I will describe the PARALEX Open QA system, which is the first system to learn a question interpretation function from the WikiAnswers paraphrase data, and also the first system to answer questions over an extracted KB. Finally, Chapter 5 describes the OQA Open QA system, which achieves higher accuracy than PARALEX and is able to leverage both extracted and curated knowledge to answer questions.

Chapter 3

IDENTIFYING RELATION PHRASES IN TEXT

Typically, information extraction systems learn an extractor for each target relation from labeled training examples (Kim and Moldovan, 1993; Riloff, 1996; Soderland, 1999). This approach to extraction does not scale to the needs of Open QA, where the target relationships are unknown in advance. Open IE solves this problem by identifying *relation phrases*—phrases that denote relations in English sentences (Banko et al., 2007). The automatic identification of relation phrases enables the extraction of *arbitrary* relations from sentences, obviating the restriction to a pre-specified vocabulary.

Open IE systems have achieved a notable measure of success on massive, open-domain corpora drawn from the web, Wikipedia, and elsewhere. (Banko et al., 2007; Wu and Weld, 2010; Zhu et al., 2009). The output of Open IE systems has been used to support tasks like learning selectional preferences (Ritter et al., 2010), acquiring common sense knowledge (Lin et al., 2010), and recognizing entailment (Schoenmackers et al., 2010; Berant et al., 2011). In addition, Open IE extractions have been mapped onto existing ontologies (Soderland et al., 2010).

We have observed that two types of errors are frequent in the output of Open IE systems such as TextRunner and woe: *incoherent extractions* and *uninformative extractions*.

Incoherent extractions are cases where the extracted relation phrase has no meaningful interpretation (see Table 3.1 for examples). Incoherent extractions arise because the learned extractor makes a sequence of decisions about whether to include each word in the relation phrase, often resulting in incomprehensible predictions. To solve this problem, we introduce a syntactic constraint: every multi-word relation phrase must begin with a verb, end with a preposition, and be a contiguous sequence of words in the sentence. Thus, the identification of a relation phrase is made in one fell swoop instead of on the basis of multiple, word-by-word decisions.

Uninformative extractions are extractions that omit critical information. For example, consider the sentence "Faust made a deal with the devil." Previous Open IE systems return the uninformative `(Faust, made, a deal)` instead of `(Faust, made a deal with, the devil)`. This type of error is caused by improper handling of relation phrases that are expressed by a combination of a verb with a noun, such as light verb constructions (LVCs). An LVC is a multi-word expression composed of a verb and a noun, with the noun carrying the semantic content of the predicate (Grefenstette and Teufel, 1995; Stevenson et al., 2004; Allerton, 2002). Table 3.2 illustrates the wide range of relations expressed this way, which are not captured by existing open extractors. Our syntactic constraint leads the extractor to include nouns in the relation phrase, solving this problem.

Although the syntactic constraint reduces incoherent and uninformative extractions, it allows overly-specific relation phrases such as `is offering only modest greenhouse gas reduction targets at`. To avoid overly-specific relation phrases, we introduce an intuitive statistical constraint: a binary relation phrase ought to appear with at least a minimal number of distinct argument pairs in a large corpus.

In summary, this chapter articulates two simple and powerful constraints on how binary relationships are expressed via verbs in English sentences, and implements them in the ReVerb Open IE system. We release ReVerb and the data used in our experiments to the research community.

The rest of the paper is organized as follows. Section 3.1 analyzes previous work. Section 3.2 defines our constraints precisely. Section 3.3 describes ReVerb, our implementation of the constraints. Section 3.4 reports on our experimental results. Section 3.4.5 concludes with a summary and discussion subsequent work that builds on top of ReVerb.

3.1 Previous Work

Open IE systems like TextRunner (Banko et al., 2007), WOE^{pos}, and WOE^{parse} (Wu and Weld, 2010) focus on extracting binary relations of the form (arg1, relation phrase, arg2) from text. These systems all use the following three-step method:

Sentence	**Incoherent Relation**
The guide *contains* dead links and *omits* sites.	`contains omits`
The Mark 14 *was central* to the *torpedo* scandal of the fleet.	`was central torpedo`
They *recalled* that Nungesser *began* his career as a precinct leader.	`recalled began`

Table 3.1: Examples of incoherent extractions. Incoherent extractions make up approximately 13% of TextRunner's output, 15% of WOE^{pos}'s output, and 30% of WOE^{parse}'s output.

`is`	`is an album by, is the author of, is a city in`
`has`	`has a population of, has a Ph.D. in, has a cameo in`
`made`	`made a deal with, made a promise to`
`took`	`took place in, took control over, took advantage of`
`gave`	`gave birth to, gave a talk at, gave new meaning to`
`got`	`got tickets to, got a deal on, got funding from`

Table 3.2: Examples of uninformative relation phrases (left) and their completions (right). Uninformative relation phrases occur in approximately 4% of WOE^{parse}'s output, 6% of WOE^{pos}'s output, and 7% of TextRunner's output.

1. **Label:** Sentences are automatically labeled with extractions using heuristics or distant supervision.

2. **Learn:** A relation phrase extractor is learned using a sequence-labeling graphical model (*e.g.*, CRF).

3. **Extract:** the system takes a sentence as input, identifies a candidate pair of NP arguments (arg1, arg2) from the sentence, and then uses the learned extractor to label each word between the two arguments as part of the relation phrase or not.

The extractor is applied to the successive sentences in the corpus, and the resulting extractions are collected.

This method faces several challenges. First, the training phase requires a large number of labeled training examples (*e.g.*, 200,000 heuristically-labeled sentences for TextRunner and 300,000 for WOE). Heuristic labeling of examples obviates hand labeling but results in noisy labels and distorts the distribution of examples. Second, the extraction step is posed as a sequence-labeling problem, where each word is assigned its own label. Because each assignment is uncertain, the likelihood that the extracted relation phrase is flawed increases with the length of the relation phrase. Finally, the extractor chooses an extraction's arguments heuristically, and cannot backtrack over this choice. This is problematic when a word that belongs in the relation phrase is chosen as an argument (for example, "deal" from the "made a deal with" sentence).

Because of the feature sets utilized in previous work, the learned extractors ignore both "holistic" aspects of the relation phrase (*e.g.*, is it contiguous?) as well as statistical aspects (*e.g.*, how many instances of this relation are there?). Thus, as we show in Section 3.4, systems such as TextRunner are unable to learn the constraints embedded in ReVerb. Of course, a learning system, utilizing a different hypothesis space, and an appropriate set of training examples, *could* potentially learn and refine the constraints in ReVerb. This is a topic for future work.

The first Open IE system was TextRunner (Banko et al., 2007), which used a Naive Bayes model with unlexicalized POS and NP-chunk features, trained using examples heuris-

tically generated from the Penn Treebank. Subsequent work showed that utilizing a linear-chain CRF (Banko and Etzioni, 2008) or Markov Logic Network (Zhu et al., 2009) can lead to improved extraction. The WOE systems introduced by Wu and Weld make use of Wikipedia as a source of training data for their extractors, which leads to further improvements over TEXTRUNNER (Wu and Weld, 2010). Wu and Weld also show that dependency parse features result in a dramatic increase in precision and recall over shallow linguistic features, but at the cost of extraction speed.

Other approaches to large-scale IE have included Preemptive IE (Shinyama and Sekine, 2006), On-Demand IE (Sekine, 2006), and weak supervision for IE (Mintz et al., 2009; Hoffmann et al., 2010). Preemptive IE and On-Demand IE avoid relation-specific extractors, but rely on document and entity clustering, which is too costly for web-scale IE. Weakly supervised methods use an existing ontology to generate training data for learning relation-specific extractors. While this allows for learning relation-specific extractors at a larger scale than what was previously possible, the extractions are still restricted to a specific ontology.

Many systems have used syntactic patterns based on verbs to extract relation phrases, usually relying on a full dependency parse of the input sentence (Lin and Pantel, 2001; Stevenson, 2004; Specia and Motta, 2006; Kathrin Eichler and Neumann, 2008). Our work differs from these approaches by focusing on relation phrase patterns expressed in terms of POS tags and NP chunks, instead of full parse trees. Banko and Etzioni (Banko and Etzioni, 2008) showed that a small set of POS-tag patterns cover a large fraction of relationships in English, but never incorporated the patterns into an extractor. This paper reports on an improved model of binary relation phrases, which increases the recall of the Banko-Etzioni model (see Section 3.2.3). Further, while previous work in Open IE has mainly focused on syntactic patterns for relation extraction, we introduce a statistical constraint that boosts precision and recall.

Finally, Open IE is closely related to semantic role labeling (SRL) (Punyakanok et al., 2008; Toutanova et al., 2008) in that both tasks extract relations and arguments from sentences. However, SRL systems traditionally rely on syntactic parsers, which makes them susceptible to parser errors and slower than Open IE systems such as REVERB. This difference is particularly important when operating on the web corpus due to its size and

$$V \mid VP \mid VW^*P$$

V = verb particle? adv?

W = (noun | adj | adv | pron | det)

P = (prep | particle | inf. marker)

Figure 3.1: A simple part-of-speech-based regular expression reduces the number of incoherent extractions like "was central torpedo" and covers relations expressed via light verb constructions like "gave a talk at."

heterogeneity. Finally, SRL requires hand-constructed semantic resources like Propbank and Framenet (Martha and Palmer, 2002; Baker et al., 1998) as input. In contrast, Open IE systems require no relation-specific training data. ReVerb, in particular, relies on its explicit statistical and syntactic constraints, which have no correlate in SRL systems. For a more detailed comparison of SRL and Open IE, see (Christensen et al., 2010).

3.2 Constraints on Relation Phrases

In this section we introduce two constraints on relation phrases: a syntactic constraint and a statistical constraint.

3.2.1 Syntactic Constraint

The syntactic constraint serves two purposes. First, it eliminates incoherent extractions, and second, it reduces uninformative extractions by capturing relation phrases expressed by a verb-noun combination, including light verb constructions.

The syntactic constraint requires the relation phrase to match the POS tag pattern shown in Figure 3.1. The pattern limits relation phrases to be either a verb (*e.g.*, "invented"), a verb followed immediately by a preposition (*e.g.*, "located in"), or a verb followed by nouns, adjectives, or adverbs ending in a preposition (*e.g.*, "has atomic weight of"). If there are multiple possible matches in a sentence for a single verb, the longest possible match is chosen. Finally, if the pattern matches multiple adjacent sequences, we merge them into a single relation phrase (*e.g.*, "wants to extend"). This refinement enables

the model to readily handle relation phrases containing multiple verbs. A consequence of this pattern is that the relation phrase must be a contiguous span of words in the sentence.

The syntactic constraint eliminates the incoherent relation phrases returned by existing systems. For example, given the sentence

> Extendicare agreed to buy Arbor Health Care for about US $432 million in cash and assumed debt.

TextRunner returns the extraction `(Arbor Health Care, for assumed, debt)`. The phrase `for assumed` is not a valid relation phrase: it begins with a preposition and splices together two distant words in the sentence. The syntactic constraint prevents this type of error by simply restricting relation phrases to match the pattern in Figure 3.1.

The syntactic constraint reduces uninformative extractions by capturing relation phrases expressed via LVCs. For example, the POS pattern matched against the sentence "Faust made a deal with the Devil," would result in the relation phrase "made a deal with," instead of the uninformative "made."

Finally, we require the relation phrase to appear between its two arguments in the sentence. This is a common constraint that has been implicitly enforced in other open extractors.

3.2.2 Statistical Constraint

While the syntactic constraint greatly reduces uninformative extractions, it can sometimes match relation phrases that are so specific that they have only a few possible instances, even in a web-scale corpus. Consider the sentence:

> The Obama administration is offering only modest greenhouse gas reduction targets at the conference.

The POS pattern will match the phrase:

$$\text{is offering only modest greenhouse gas reduction targets at} \tag{3.1}$$

Thus, there are phrases that satisfy the syntactic constraint, but are not relational.

To overcome this limitation, we introduce a statistical constraint that is used to separate valid relation phrases from overspecified relation phrases, like the example in (3.1). The constraint is based on the intuition that a valid relation phrase should take many distinct arguments in a large corpus. The phrase in (3.1) is specific to the argument pair `(Obama administration, conference)`, so it is unlikely to represent a *bona fide* relation. We describe the implementation details of the statistical constraint in Section 3.3.

3.2.3 Limitations

Our constraints represent an idealized model of relation phrases in English. This raises the question: How much recall is lost due to the constraints?

To address this question, we analyzed Wu and Weld's set of 300 sentences from a set of random web pages, manually identifying all verb-based relationships between noun phrase pairs. This resulted in a set of 327 relation phrases. For each relation phrase, we checked whether it satisfies our constraints. We found that 85% of the relation phrases do satisfy the constraints. Of the remaining 15%, we identified some of the common cases where the constraints were violated, summarized in Table 3.3.

Many of the example relation phrases shown in Table 3.3 involve long-range dependencies between words in the sentence. These types of dependencies are not easily representable using a pattern over POS tags. A deeper syntactic analysis of the input sentence would provide a much more general language for modeling relation phrases. For example, one could create a model of relations expressed in terms of dependency parse features that would capture the non-contiguous relation phrases in Table 3.3. Previous work has shown that dependency paths do indeed boost the recall of relation extraction systems (Wu and Weld, 2010; Mintz et al., 2009). While using dependency path features allows for a more flexible model of relations, it increases processing time, which is problematic for web-scale extraction. Further, we have found that this increased recall comes at the cost of lower precision on web text (see Section 3.4).

The results in Table 3.3 are similar to Banko and Etzioni's findings that a set of eight POS patterns cover a large fraction of binary verbal relation phrases. However, their analysis

	Binary Verbal Relation Phrases
85%	Satisfy Constraints
8%	Non-Contiguous Phrase Structure
	Coordination: X is produced and maintained by Y
	Multiple Args: X was founded in 1995 by Y
	Phrasal Verbs: X turned Y off
4%	Relation Phrase Not Between Arguments
	Intro. Phrases: Discovered by Y, X ...
	Relative Clauses: ... the Y that X discovered
3%	Do Not Match POS Pattern
	Interrupting Modifiers: X has a lot of faith in Y
	Infinitives: X to attack Y

Table 3.3: Approximately 85% of the binary verbal relation phrases in a sample of web sentences satisfy our constraints.

was based on a set of sentences known to contain either a company acquisition or birthplace relationship, while our results are on a random sample of web sentences. We applied Banko and Etzioni's verbal patterns to our random sample of 300 web sentences, and found that they cover approximately 69% of the relation phrases in the corpus. The gap in recall between this and the 85% shown in Table 3.3 is due to LVC relation phrases ("made a deal with") and phrases containing multiple verbs ("refuses to return to"), which their patterns do not cover.

In sum, our model is by no means complete. However, we have empirically shown that the majority of binary verbal relation phrases in a sample of web sentences are captured by our model. By focusing on this subset of language, our model can be used to perform Open IE at higher precision than before.

3.3 ReVerb

This section introduces ReVerb, a novel open extractor based on the constraints defined in the previous section. ReVerb first identifies relation phrases that satisfy the syntactic and statistical constraints, and then finds a pair of NP arguments for each identified relation phrase. The resulting extractions are then assigned a confidence score using a logistic regression classifier.

This algorithm differs in three important ways from previous methods (Section 3.1). First, the relation phrase is identified "holistically" rather than word-by-word. Second, potential phrases are filtered based on statistics over a large corpus (the implementation of our statistical constraint). Finally, ReVerb is "relation first" rather than "arguments first", which enables it to avoid a common error made by previous methods—confusing a noun in the relation phrase for an argument, *e.g.* the noun "deal" in "made a deal with."

3.3.1 Extraction Algorithm

ReVerb takes as input a POS-tagged and NP-chunked sentence and returns a set of (x, r, y) extraction triples.[1] Given an input sentence s, ReVerb uses the following extraction algo-

[1] ReVerb uses OpenNLP for POS tagging and NP chunking: `http://opennlp.sourceforge.net/`

rithm:

1. **Relation Extraction:** For each verb v in s, find the longest sequence of words r_v such that (1) r_v starts at v, (2) r_v satisfies the syntactic constraint, and (3) r_v satisfies the statistical constraint. If any pair of matches are adjacent or overlap in s, merge them into a single match.

2. **Argument Extraction:** For each relation phrase r identified in Step 1, find the nearest noun phrase x to the left of r in s such that x is not a relative pronoun, Wh-adverb, or existential "there". Find the nearest noun phrase y to the right of r in s. If such an (x, y) pair could be found, return (x, r, y) as an extraction.

We check whether a candidate relation phrase r_v satisfies the syntactic constraint by matching it against the regular expression in Figure 3.1.

To determine whether r_v satisfies the statistical constraint, we use a large dictionary D of relation phrases that are known to take many distinct arguments. In an offline step, we construct D by finding all matches of the POS pattern in a corpus of 500 million web sentences. For each matching relation phrase, we heuristically identify its arguments (as in Step 2 above). We set D to be the set of all relation phrases that take at least k distinct argument pairs in the set of extractions. In order to allow for minor variations in relation phrases, we normalize each relation phrase by removing inflection, auxiliary verbs, adjectives, and adverbs. Based on experiments on a held-out set of sentences, we found that a value of $k = 20$ works well for filtering out overspecified relations. This results in a set of approximately 1.7 million distinct normalized relation phrases, which are stored in memory at extraction time.

As an example of the extraction algorithm in action, consider the following input sentence:

> Hudson was born in Hampstead, which is a suburb of London.

Step 1 of the algorithm identifies three relation phrases that satisfy the syntactic and statistical constraints: "was," "born in," and "is a suburb of." The first two phrases are adjacent

in the sentence, so they are merged into the single relation phrase "was born in." Step 2 then finds an argument pair for each relation phrase. For "was born in," the nearest NPs are (Hudson, Hampstead). For "is a suburb of," the extractor skips over the NP "which" and chooses the argument pair (Hampstead, London). The final output contains two extractions: `(Hudson, was born in, Hampstead)` and `(Hampstead, is a suburb of, London)`.

3.3.2 Confidence Function

The extraction algorithm in the previous section has high recall, but low precision. Like with previous open extractors, we want way to trade recall for precision by tuning a confidence threshold. We use a logistic regression classifier to assign a confidence score to each extraction, which uses the features shown in Table 3.4. All of these features are efficiently computable and depend only on POS tags and closed-class words. We trained the confidence function by manually labeling the extractions from a set of $1,000$ sentences from the web and Wikipedia as correct or incorrect.

Previous Open IE systems require labeled training data to learn a model of relations, which is then used to extract relation phrases from text. In contrast, ReVerb uses a specified model of relations for extraction, and requires labeled data only for assigning confidence scores to its extractions. Learning a confidence function is a much simpler task than learning a full model of relations, using two orders of magnitude fewer training examples than TextRunner or woe.

3.3.3 TextRunner-R

The model of relation phrases used by ReVerb is specified, but could a TextRunner-like system learn this model from training data? While it is difficult to answer such a question for all possible permutations of features sets, training examples, and learning biases, we demonstrate that TextRunner itself cannot learn ReVerb's model even when re-trained using the output of ReVerb as labeled training data. The resulting system, TextRunner-R, uses the same feature representation as TextRunner, but different parameters, and a different set of training examples.

Weight	Feature
1.16	(x, r, y) covers all words in s
0.50	The last preposition in r is "for"
0.49	The last preposition in r is "on"
0.46	The last preposition in r is "of"
0.43	$len(s) \leq 10$ words
0.43	There is a WH-word to the left of r
0.42	r matches VW*P from Figure 3.1
0.39	The last preposition in r is "to"
0.25	The last preposition in r is "in"
0.23	10 words $< len(s) \leq 20$ words
0.21	s begins with x
0.16	y is a proper noun
0.01	x is a proper noun
-0.30	There is an NP to the left of x in s
-0.43	20 words $< len(s)$
-0.61	r matches V from Figure 3.1
-0.65	There is a preposition to the left of x in s
-0.81	There is an NP to the right of y in s
-0.93	Coord. conjunction to the left of r in s

Table 3.4: ReVerb uses these features to assign a confidence score to an extraction (x, r, y) from a sentence s using a logistic regression classifier.

To generate positive instances, we ran ReVerb on the Penn Treebank, which is the same dataset that TextRunner is trained on. To generate negative instances from a sentence, we took each noun phrase pair in the sentence that does not appear as arguments in a ReVerb extraction. This process resulted in a set of 67,562 positive instances, and 356,834 negative instances. We then passed these labeled examples to TextRunner's training procedure, which learns a linear-chain CRF using closed-class features like POS tags, capitalization, punctuation, *etc.*TextRunner-R uses the argument-first extraction algorithm described in Section 3.1.

3.4 Experiments

We compare ReVerb to the following systems:

- ReVerb$^{\neg stat}$ - The ReVerb system described in the previous section, but without the statistical constraint. ReVerb$^{\neg stat}$ uses the same confidence function as ReVerb.

- TextRunner - Banko and Etzioni's 2008 extractor, which uses a second order linear-chain CRF trained on extractions heuristically generated from the Penn Treebank. TextRunner uses shallow linguistic features in its CRF, which come from the same POS tagger and NP-chunker that ReVerb uses.

- TextRunner-R - Our modification to TextRunner, which uses the same extraction code, but with a model of relations trained on ReVerb extractions.

- WOE^{pos} - Wu and Weld's modification to TextRunner, which uses a model of relations learned from extractions heuristically generated from Wikipedia.

- WOE^{parse} - Wu and Weld's parser-based extractor, which uses a large dictionary of dependency path patterns learned from heuristic extractions generated from Wikipedia.

Each system is given a set of sentences as input, and returns a set of binary extractions as output. We created a test set of 500 sentences sampled from the web, using Yahoo's

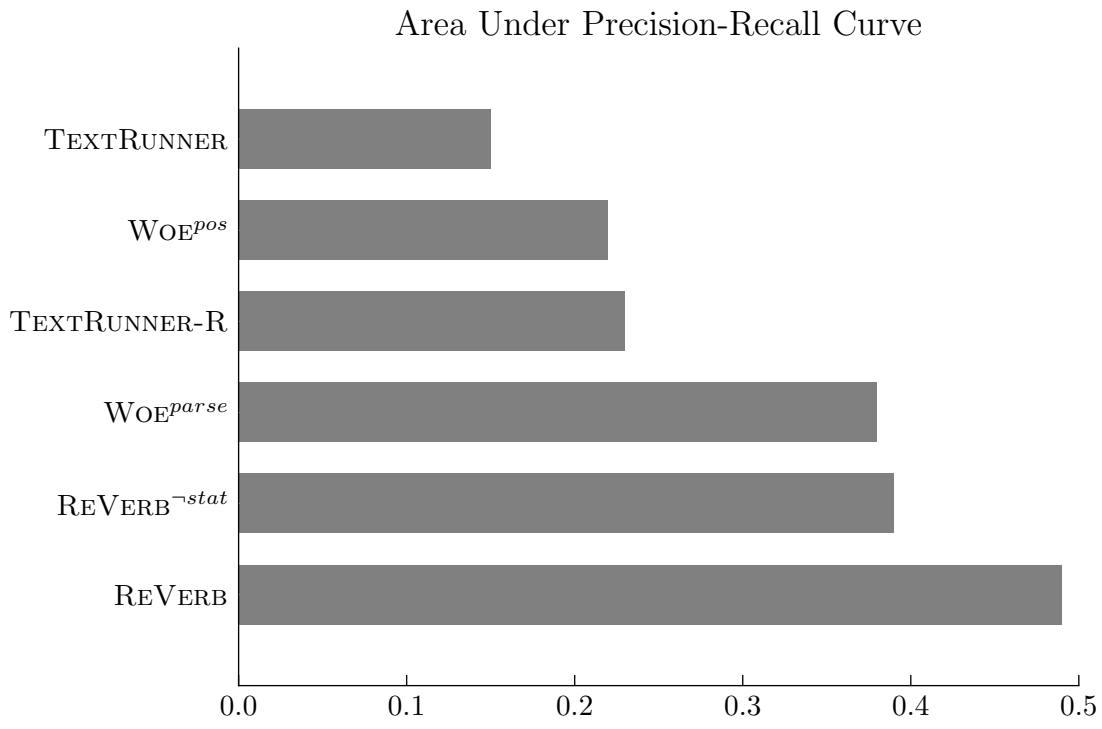


Figure 3.2: ReVerb outperforms state-of-the-art open extractors, with an AUC more than twice that of TextRunner or WOE^{pos}, and 38% higher than WOE^{parse}.

random link service.[2] After running each extractor over the input sentences, two human judges independently evaluated each extraction as correct or incorrect. The judges reached agreement on 86% of the extractions, with an agreement score of $\kappa = 0.68$. We report results on the subset of the data where the two judges concur.

The judges labeled uninformative extractions conservatively. That is, if critical information was dropped from the relation phrase but included in the second argument, it is labeled correct. For example, both the extractions `(Ackerman, is a professor of, biology)` and `(Ackerman, is, a professor of biology)` are considered correct.

Each system returns confidence scores for its extractions. For a given threshold, we can measure the precision and recall of the output. Precision is the fraction of returned

[2] `http://random.yahoo.com/bin/ryl`

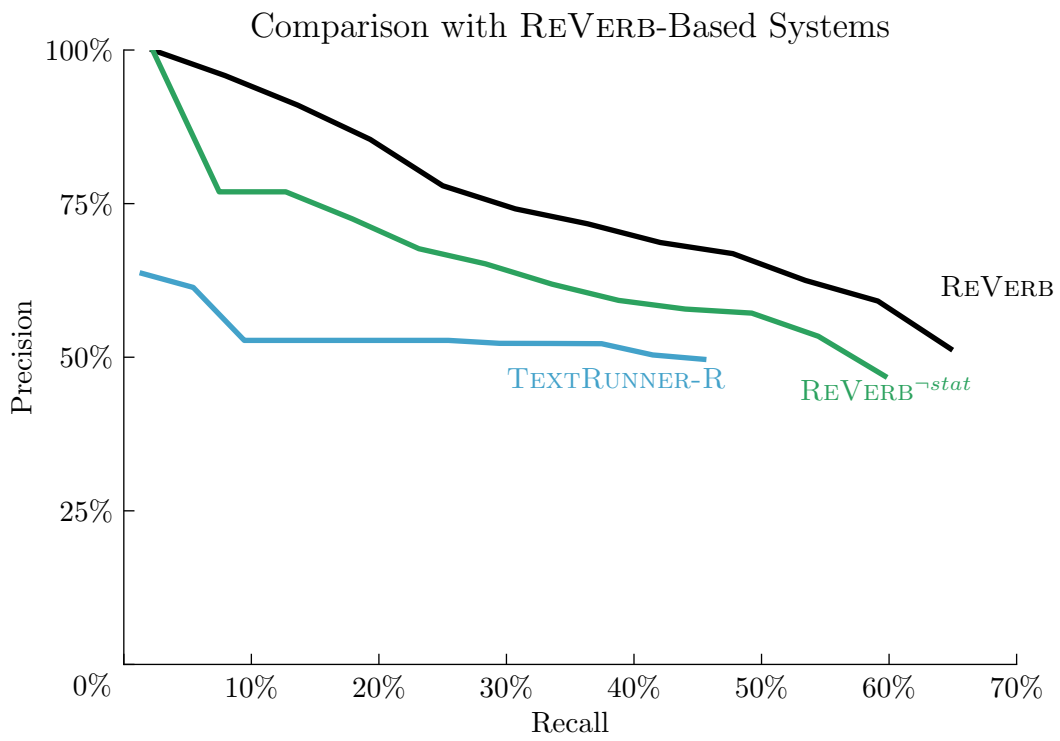


Figure 3.3: The statistical constraint gives ReVerb a boost in precision and recall over ReVerb$^{\neg stat}$. TextRunner-R is unable to learn the model used by ReVerb, which results in lower precision and recall.

extractions that are correct. Recall is the fraction of correct extractions in the corpus that are returned. We use the total number of extractions labeled as correct by the judges as our measure of recall for the corpus. In order to avoid double-counting, we treat extractions that differ superficially (*e.g.*, different punctuation or dropping inessential modifiers) as a single extraction. We compute a precision-recall curve by varying the confidence threshold, and then compute the area under the curve (AUC).

3.4.1 Results

Figure 3.2 shows the AUC of each system. ReVerb achieves an AUC that is 30% higher than WOE^{parse} and is more than double the AUC of WOE^{pos} or TextRunner. The statistical constraint provides a boost in performance, with ReVerb achieving an AUC 23% higher than ReVerb$^{\neg stat}$. ReVerb proves to be a useful source of training data, with TextRunner-R having an AUC 71% higher than TextRunner and performing on par with WOE^{pos}. From the training data, TextRunner-R was able to learn a model that predicts contiguous relation phrases, but still returned incoherent relation phrases (*e.g.*, starting with a preposition) and overspecified relation phrases. These errors are due to TextRunner-R overfitting the training data and not having access to the statistical constraint.

Figure 3.3 shows the precision-recall curves of the systems introduced in this paper. TextRunner-R has much lower precision than ReVerb and ReVerb$^{\neg stat}$ at all levels of recall. The statistical constraint gives ReVerb a boost in precision over ReVerb$^{\neg stat}$, reducing overspecified extractions from 20% of ReVerb$^{\neg stat}$'s output to 1% of ReVerb's. The statistical constraint also boosts recall over ReVerb$^{\neg stat}$, since ReVerb is able to find a correct relation phrase where ReVerb$^{\neg stat}$ finds an overspecified one.

Figure 3.4 shows the precision-recall curves of ReVerb and the external systems. ReVerb has much higher precision than the other systems at nearly all levels of recall. In particular, more than 30% of ReVerb's extractions are at precision 0.8 or higher, compared to virtually none for the other systems. WOE^{parse} achieves a slightly higher recall than ReVerb (0.62 versus 0.64), but at the cost of lower precision.

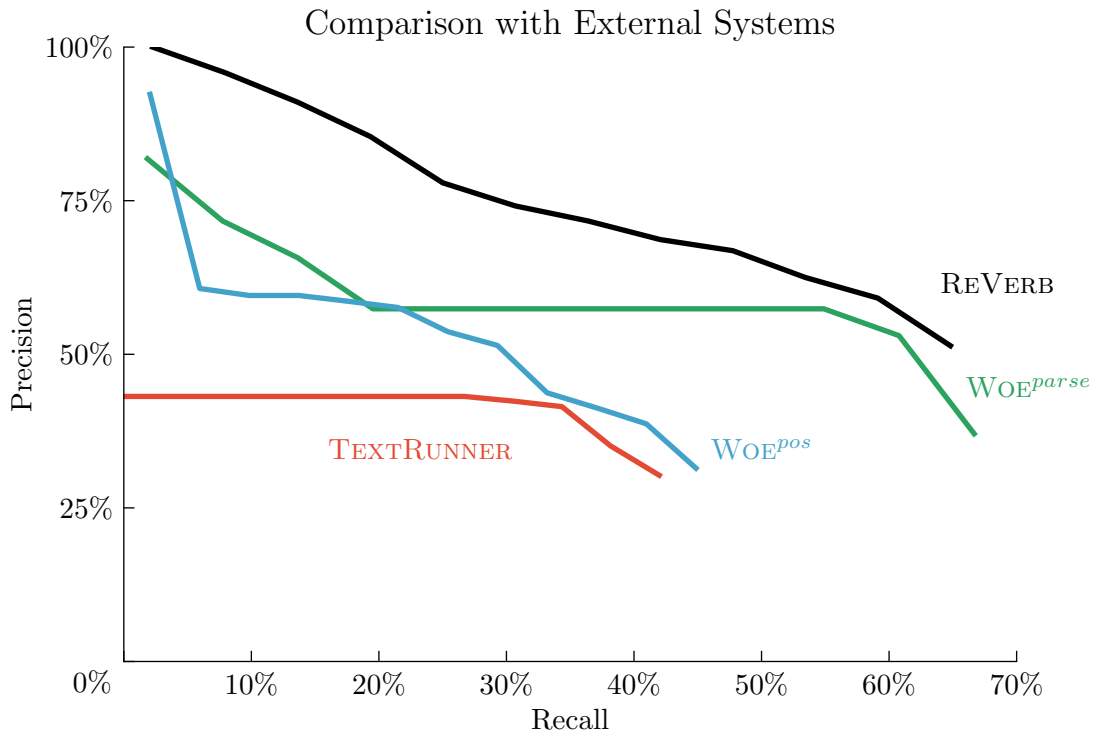


Figure 3.4: ReVerb achieves higher precision than state-of-the-art Open IE systems, and comparable recall to WOE^{parse}.

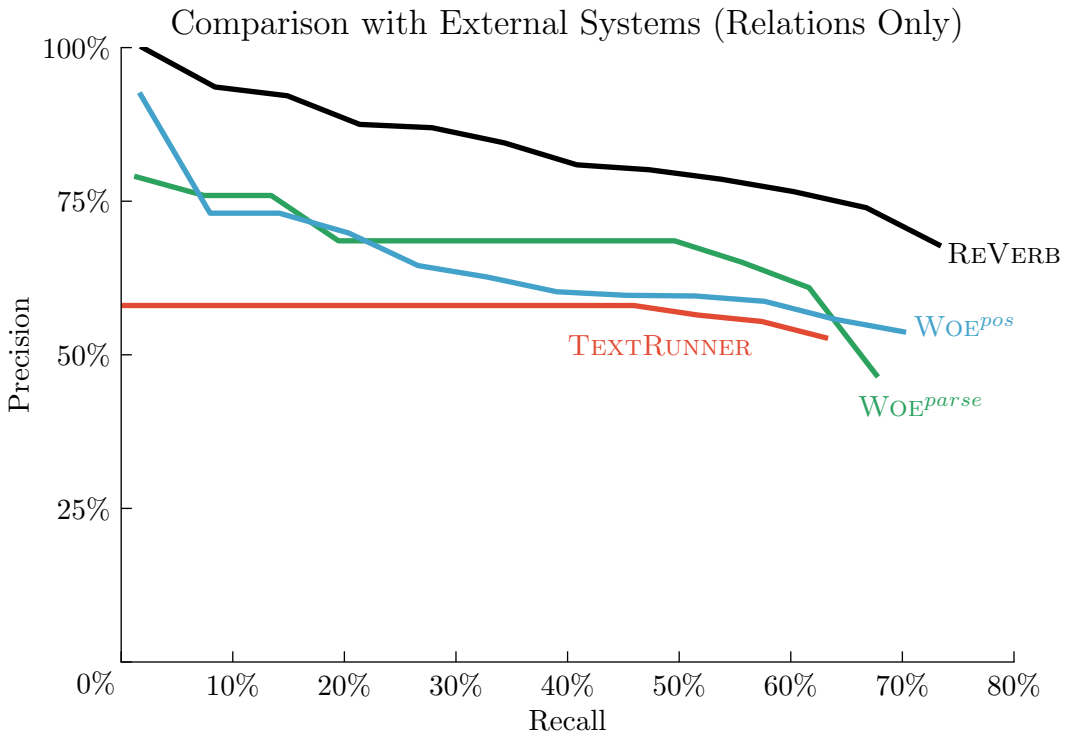


Figure 3.5: On the subtask of identifying relations phrases, ReVerb is able to achieve even higher precision and recall than other systems.

In order to highlight the role of the relational model of each system, we also evaluate their performance on the subtask of extracting just the relation phrases from the input text. Figure 3.5 shows the precision-recall curves for each system on the relation phrase-only evaluation. In this case, ReVerb has both higher precision and recall than the other systems.

ReVerb's biggest improvement came from the elimination of incoherent extractions. Incoherent extractions were a large fraction of the errors made by previous systems, accounting for approximately 13% of TextRunner's extractions, 15% of WOE^{pos}'s, and 30% of WOE^{parse}'s. Uninformative extractions had a smaller effect on other systems' precision, accounting for 4% of WOE^{parse}'s extractions, 5% of WOE^{pos}'s, and 7% of TextRunner's, while only appearing in 1% of ReVerb's extractions. ReVerb's reduction in uninformative

ReVerb - Incorrect Extractions	
65%	Correct relation phrase, incorrect arguments
16%	N-ary relation
8%	Non-contiguous relation phrase
2%	Imperative verb
2%	Overspecified relation phrase
7%	Other, including POS/chunking errors

Table 3.5: The majority of the incorrect extractions returned by ReVerb are due to errors in argument extraction.

extractions resulted in a boost in recall, capturing many LVC relation phrases missed by other systems (like those shown in Table 3.2).

To test the systems' speed, we ran each extractor on a set of 100,000 sentences using a Pentium 4 machine with 4GB of RAM. The processing times were 16 minutes for ReVerb, 21 minutes for TextRunner, 21 minutes for WOE^{pos}, and 11 hours for WOE^{parse}. The times for ReVerb, TextRunner, and WOE^{pos} are all approximately the same, since they all use the same POS-tagging and NP-chunking software. WOE^{parse} processes each sentence with a dependency parser, resulting in much longer processing time. For a more in-depth analysis of ReVerb and other Open IE systems, see the work by Mesquita et al. (2013).

3.4.2 ReVerb *vs.* TextRunner-R

The experimental results show that TextRunner-R did not learn the ReVerb constraints from the training data. This raises two questions. First, can TextRunner even represent the ReVerb constraints? Second, if TextRunner can indeed represent the ReVerb constraints, why was it unable to learn them from a sample of data?

TextRunner does not use statistical features, so it cannot possibly learn the ReVerb statistical constraint from the training data. However, TextRunner *can* represent the syntactic constraint. To see why, recall that TextRunner uses a linear-chain CRF to label

	Left Label	Left POS	Right Label	Right POS
1	O-REL	Any	O-REL	Any
2	O-REL	Any	B-REL	V
3	B-REL	V	I-REL	W
4	B-REL	V	I-REL	P
5	I-REL	W	I-REL	W
6	I-REL	W	I-REL	P
7	I-REL	P	B-REL	V
8	I-REL	P	O-REL	Any

Table 3.6: A subset of TEXTRUNNER's features that emulate REVERB's VW*P syntactic constraint. Each row is a predicate over a pair of adjacent output labels and their POS tags.

a sequence of words as B-REL (the beginning word of a relation phrase), I-REL (a word inside of a relation phrase), or O-REL (a word outside of a relation phrase). To discriminate between possible output labels, TEXTRUNNER uses boolean indicator features that test whether a pair of adjacent output labels and POS tags match a particular pattern. Using the same POS notation as in Figure 3.1, an example TEXTRUNNER feature is:

$$\text{Left Label} = \text{B-REL} \wedge \text{Left POS} = \text{V} \wedge \text{Right Label} = \text{I-REL} \wedge \text{Right POS} = \text{P}$$

This feature evaluates to true when there is a pair of adjacent output labels (B-REL, I-REL) with POS tags (V, P). These features can encode the regular expression that defines REVERB's syntactic constraint. Any regular expression can be translated into a nondeterministic finite automaton (NFA), which defines a set of transitions that are allowed when matching the regular expression against an input string. For example, the regular expression VW*P can be encoded as the disjunction of the features shown in Table 3.6. Each row of Table 3.6 corresponds to a legal transition in the NFA for VW*P. For example, the input string (W, V, W, P, W) with predicted labels (O-REL, B-REL, I-REL, I-REL, O-REL) will cause features 2, 3, 6, and 8 to fire. If we assign large, positive weights to the features

in Table 3.6, then TextRunner's highest-probability prediction will agree with the regular expression VW*P.

Thus, TextRunner is capable of representing ReVerb's syntactic constraint, but failed to learn it from data. The TextRunner feature representation uses on the order of ten thousand indicator functions, only a small subset of which are necessary to encode the syntactic constraint. The CRF learner searches for a parameterization of these features that maximizes the regularized likelihood of the training data. The parameterization that maximizes this objective function does not necessarily correspond to a compact regular expression like those in Figure 3.1. One potential solution to this problem would be to bias the learner to prefer parameterizations that correspond to simple regular expressions over POS tags, which could lead to better generalization and fewer incoherent extractions.

3.4.3 ReVerb *Error Analysis*

To better understand the limitations of ReVerb, we performed a detailed analysis of its errors in precision (incorrect extractions returned by ReVerb) and its errors in recall (correct extractions that ReVerb missed).

Table 3.5 summarizes the types of incorrect extractions that ReVerb returns. We found that 65% of the incorrect extractions returned by ReVerb were cases where a relation phrase was correctly identified, but the argument-finding heuristics failed. The remaining errors were cases where ReVerb extracted an incorrect relation phrase. One common mistake that ReVerb made was extracting a relation phrase that expresses an n-ary relationship via a ditransitive verb. For example, given the sentence "I gave him 15 photographs," ReVerb extracts `(I, gave, him)`. These errors are due to the fact that ReVerb only captures binary relations.

Table 3.7 summarizes the correct extractions that were extracted by other systems and were not extracted by ReVerb. As with the false-positive extractions, the majority of false negatives (52%) were due to the argument-finding heuristics choosing the wrong arguments, or failing to extract all possible arguments (in the case of coordinating conjunctions). Other sources of failure were due to the statistical constraint either failing to filter out an over-

ReVerb - Missed Extractions	
52%	Could not identify correct arguments
23%	Relation filtered out by lexical constraint
17%	Identified a more specific relation
8%	POS/chunking error

Table 3.7: The majority of extractions that were missed by ReVerb were cases where the correct relation phrase was found, but the arguments were not correctly identified.

specified relation phrase or filtering out a valid relation phrase. These errors hurt both precision and recall, since each case results in the extractor overlooking a correct relation phrase and choosing another.

3.4.4 Evaluation At Scale

Section 3.4.1 shows that ReVerb outperforms existing Open IE systems when evaluated on a sample of sentences. Previous work has shown that the frequency of an extraction in a large corpus is useful for assessing the correctness of extractions (Downey et al., 2005). Thus, it is possible *a priori* that ReVerb's gains over previous systems will diminish when extraction frequency is taken into account.

In fact, we found that ReVerb's advantage over TextRunner when run at scale is qualitatively similar to its advantage on single sentences. We ran both ReVerb and TextRunner on Banko and Etzioni's corpus of 500 million web sentences and examined the effect of redundancy on precision.

As Downey's work predicts (Downey et al., 2005), precision increased in both systems for extractions found multiple times, compared with extractions found only once. However, ReVerb had higher precision than TextRunner at all frequency thresholds. In fact, ReVerb's frequency 1 extractions had a precision of 0.75, which TextRunner could not approach even with frequency-10 extractions, which had a precision of 0.34. Thus, ReVerb is able to return more correct extractions at a higher precision than TextRunner, even

when redundancy is taken into account.

3.4.5 Improvements and Applications

Since ReVerb was released in 2011, there have been many new Open IE systems that make improvements over ReVerb and ReVerb extractions have been used in many applications. In this section, I highlight some of these improvements and applications.

Improvements and Extensions to ReVerb

- While the work on ReVerb focused on identifying relation phrases, it did not directly address the problem of argument identification. ReVerb uses simple heuristic rules to identify the arguments of a relation, leading to many systematic errors in its extractions. Etzioni et al. introduce the ArgLearner system that provides more reliable argument identification than ReVerb's heuristics (Etzioni et al., 2011).

- As noted earlier, ReVerb only extracts binary relationships between a pair of entities. This hurts ReVerb's recall, and Table 3.5 shows that it also lowers precision. The KrakeN system (Akbik and Löser, 2012) is an Open IE system that outperforms ReVerb by specifically targeting n-ary relationships in text.

- There has been work on increasing the recall of ReVerb by using its syntactic pattern as a way to heuristically label data, and then use machine-learning techniques to generalize from it. While this approach had limited efficacy for TextRunner, researchers were able to design new systems that could leverage this heuristically labeled data (Schmitz et al., 2012; Xu et al., 2013).

- ReVerb uses a flat, POS-tag representation for its syntactic pattern. While Table 3.3 shows that many relation phrases can be captured this way, it is unable to capture relation phrases expressed as non-contiguous phrases. The Ollie (Schmitz et al., 2012) and Dep-OE systems explore using dependency-tree representations to increase recall.

- While ReVerb is limited to English text, other researchers have ported it to other languages including Korean (Kim and Lee, 2012) and Chinese (Tseng et al., 2014) using similar approaches.

- Researchers have acquired human-annotated training data for ReVerb via a mobile crowdsourcing application (Vaish et al., 2014).

Applications of ReVerb

ReVerb has also been used for other natural language processing tasks:

- The ReVerb syntactic pattern has been used as a feature for entity extraction (Ling and Weld, 2012).

- ReVerb extractions have been used extensively for the purpose of inferring entailment relationships between relation phrases (Berant et al., 2011, 2012; Zeichner, 2012) and inferring unseen relationships (Angeli and Manning, 2013; Chen and Wang, 2014).

- ReVerb extractions have also been linked to Freebase (Lin et al., 2012) and then used to bootstrap semantic lexicons for large-scale QA systems (Cai and Yates, 2013; Berant et al., 2013).

3.5 Conclusion

This Chapter presented ReVerb, a new technique for Open Information Extraction that proved to be faster, simpler, and more accurate than existing systems. The output of ReVerb is used in the rest of this disseration as a source of knowledge for the Open QA systems Paralex (Chapter 4) and oqa (Chapter 5).

Chapter 4

PARAPHRASE-DRIVEN LEARNING FOR OPEN QUESTION ANSWERING

In this chapter, we present work on the question interpretation problem for Open QA. The PARALEX system described in this chapter is the first system to perform Open QA over an extracted knowledge base. PARALEX uses paraphrases from WikiAnswers to learn a function from questions to knowledge-base queries.

Table 4.1 shows example WikiAnswers paraphrase clusters for a set of factual questions. Such data provides strong signal for learning about lexical variation, but there are a number of challenges. Given that the data is community-authored, it will inevitably be incomplete, contain incorrectly tagged paraphrases, non-factual questions, and other sources of noise.

Our core contribution is a new learning approach that scalably sifts through this paraphrase noise, learning to answer a broad class of factual questions. We focus on answering open-domain questions that can be answered with single-relation queries, *e.g.* all of the paraphrases of "Who wrote Winnie the Pooh?" and "What cures a hangover?" in Table 4.1. The algorithm answers such questions by mapping them to executable queries over a tuple store containing relations such as `(milne, authored, winnie the pooh)` and `(bloody mary, treats, hangover symptoms)`.

The approach automatically induces lexical structures, which are combined to build queries for unseen questions. It learns lexical equivalences for relations (*e.g.*, "wrote," "authored," and "creator"), entities (*e.g.*, "Winnie the Pooh" or "Pooh Bear"), and question templates (*e.g.*, "Who `r` the `e` books?" or "Who is the `r` of `e`?"). Crucially, the approach does not require any explicit labeling of the questions in our paraphrase corpus. Instead, we use 16 seed question templates and string-matching to find high-quality queries for a small subset of the questions. The algorithm uses learned word alignments to aggressively generalize the seeds, producing a large set of possible lexical equivalences. We then learn a

Who wrote the Winnie the Pooh books?
Who is the author of winnie the pooh?
What was the name of the authur of winnie the pooh?
Who wrote the series of books for Winnie the poo?
Who wrote the children's storybook 'Winnie the Pooh'?
Who is poohs creator?
What relieves a hangover?
What is the best cure for a hangover?
The best way to recover from a hangover?
Best remedy for a hangover?
What takes away a hangover?
How do you lose a hangover?
What helps hangover symptoms?
What are social networking sites used for?
Why do people use social networking sites worldwide?
Advantages of using social network sites?
Why do people use social networks a lot?
Why do people communicate on social networking sites?
What are the pros and cons of social networking sites?
How do you say Santa Claus in Sweden?
Say santa clause in sweden?
How do you say santa clause in swedish?
How do they say santa in Sweden?
In Sweden what is santa called?
Who is sweden santa?

Table 4.1: Examples of paraphrase clusters from the WikiAnswers corpus. Within each cluster, there is a wide range of syntactic and lexical variations.

linear ranking model to filter the learned lexical equivalences, keeping only those that are likely to answer questions well in practice.

Experimental results on 18 million paraphrase pairs gathered from WikiAnswers. demonstrate the effectiveness of the overall approach. We performed an end-to-end evaluation against a knowledge base of 15 million facts automatically extracted from web text using ReVerb. On known-answerable questions, the approach achieved 42% recall, with 77% precision, more than quadrupling the recall over a baseline system.

We make the following contributions in this chapter:

- We introduce Paralex, an end-to-end Open QA system.
- We describe scalable learning algorithms that induce general question templates and lexical variants of entities and relations. These algorithms require no manual annotation and can be applied to large, noisy knowledge bases of relational triples.
- We evaluate Paralex on the end-task of answering questions from WikiAnswers using a knowledge base of ReVerb extractions, and show that it outperforms baseline systems.
- We release our learned lexicon and question-paraphrase dataset to the research community, available at `http://openie.cs.washington.edu`.

4.1 Overview of the Approach

In this section, we give a high-level overview of the rest of the chapter.

Problem Our goal is to learn a function that will map a natural language question x to a query z over a knowledge base K. The knowledge base K is a collection of assertions in the form (e_1, r, e_2) where r is a binary relation from a vocabulary R, and e_1 and e_2 are entities from a vocabulary E. We assume that the elements of R and E are human-interpretable strings like `population` or `new york`. In our experiments, R and E contain millions of entries representing ambiguous and overlapping concepts. The knowledge base is equipped with a simple interface that accepts queries in the form $(?, r, e_2)$ or $(e_1, r, ?)$. When executed,

these queries return all entities e that satisfy the given relationship. Thus, our task is to find the query z that best captures the semantics of the question x.

Model The question answering model includes a lexicon and a linear ranking function. The lexicon L associates natural language patterns to knowledge base concepts, thereby defining the space of queries that can be derived from the input question (see Table 4.2). Lexical entries can pair strings with knowledge base entities ("nyc" and `new york`), strings with knowledge base relations ("big" and `population`), or question patterns with templated knowledge base queries ("How `r` is `e`?" and `(?, r, e)`). We describe this model in more detail in Section 4.2.

Learning The learning algorithm induces a lexicon L and estimates the weights $\mathbf{w}$ of the linear ranking function. We learn L by bootstrapping from an initial seed lexicon L_0 over a corpus of question paraphrases $\mathcal{C} = \{(x, x') : x' \text{ is a paraphrase of } x\}$, like the examples in Table 4.1. We estimate $\mathbf{w}$ by using the initial lexicon to automatically label queries in the paraphrase corpus, as described in Section 4.3.2. The final result is a scalable learning algorithm that requires no manual annotation of questions.

Evaluation In Section 4.6, we evaluate our system against baseline systems on the end-task of question answering against a large knowledge base of facts extracted from the web using ReVerb. We use held-out, known-answerable questions from WikiAnswers as a test set.

4.2 Question Interpretation Model

To answer questions, we must find the best query for a given natural language question.

4.2.1 Lexicon and Derivations

To define the space of possible queries, Paralex uses a lexicon L that encodes mappings from natural language to knowledge base concepts (entities, relations, and queries). Each entry in L is a pair (p, c) where p is a pattern and c is an associated knowledge base concept. Table 4.2 gives examples of the entry types in L: entity, relation, and question patterns.

Entry Type	Natural Language Pattern	Knowledge Base Concept
Entity	"nyc"	`new york`
Relation	"big"	`population`
Question (1-Arg.)	"How big is `e`?"	`(?, population, e)`
Question (2-Arg.)	"How `r` is `e`?"	`(?, r, e)`

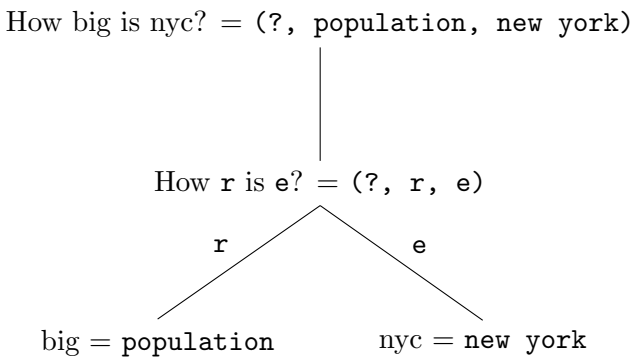
Table 4.2: Example lexical entries.

Entity patterns match a contiguous string of words and are associated with some knowledge base entity $e \in E$.

Relation patterns match a contiguous string of words and are associated with a relation $r \in R$ and an argument ordering (*e.g.* the string "child" could be modeled as either `parent of` or `child of` with opposite argument ordering).

Question patterns match an entire question string, with gaps that recursively match an entity or relation patterns. Question patterns are associated with a templated knowledge base query, where the values of the variables are determined by the matched entity and relation patterns. A question pattern may be **1-Argument**, with a variable for an entity pattern, or **2-Argument**, with variables for an entity pattern and a relation pattern. A 2-argument question pattern may also invert the argument order of the matched relation pattern, *e.g.* "Who `r e`?" may have the opposite argument order of "Who did `e r`?"

The lexicon is used to generate a derivation y from an input question x to a knowledge base query z. For example, the entries in Table 4.2 can be used to make the following derivation from the question "How big is nyc?" to the query `(?, population, new york)`:



This derivation proceeds in two steps: first matching a question form like "How `r` is `e`?" and then mapping "big" to `population` and "nyc" to `new york`. Factoring the derivation this way allows the lexical entries for "big" and "nyc" to be reused in semantically equivalent variants like "nyc how big is it?" or "Approximately how big is nyc?" This factorization helps the system generalize to novel questions that do not appear in the training set.

We model a derivation as a set of (p_i, c_i) pairs, where each p_i matches a substring of x, the substrings cover all words in x, and the knowledge base concepts c_i compose to form z. Derivations are rooted at either a 1-argument or 2-argument question entry and have entity or relation entries as leaves.

4.2.2 Linear Ranking Function

In general, multiple queries may be derived from a single input question x using a lexicon L. Many of these derivations may be incorrect due to noise in L. Given a question x, we consider all derivations y and score them with $\mathbf{w} \cdot \mathbf{f}(x, y)$, where $\mathbf{f}(x, y)$ is a n-dimensional feature representation and $\mathbf{w}$ is a n-dimensional weight vector. Let $\mathsf{Derivs}(x, L)$ be the set of all derivations y that can be generated from x using L. The best derivation $y^*(x)$ according to the model $(\mathbf{w}, L)$ is given by:

$$y^*(x) = \underset{y \in \mathsf{Derivs}(x,L)}{\arg\max} \; \mathbf{w} \cdot \mathbf{f}(x, y)$$

The best query $z^*(x)$ can be computed directly from the derivation $y^*(x)$.

Computing the set $\mathsf{Derivs}(x, L)$ involves finding all 1-Argument and 2-Argument question patterns that match x, and then enumerating all possible knowledge base concepts that

match entity and relation strings. When the knowledge base and lexicon are large, this becomes intractable. We prune $\mathsf{Derivs}(x, L)$ using the model weights $\mathbf{w}$ by only considering the N-best question patterns that match x, before additionally enumerating any relations or entities.

For the end-to-end QA task, we return a ranked list of answers from the k highest scoring queries. We score an answer a with the highest score of all derivations that generate a query with answer a.

4.3 Learning

PARALEX uses a two-part learning algorithm; it first induces an overly general lexicon (Section 4.3.1) and then learns to score derivations to increase accuracy (Section 4.3.2). Both algorithms rely on an initial seed lexicon, which we describe in Section 4.5.4.

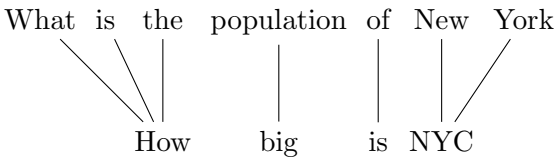
4.3.1 Lexical Learning

The lexical learning algorithm constructs a lexicon L from a corpus of question paraphrases $\mathcal{C} = \{(x, x') : x' \text{ is a paraphrase of } x\}$, where we assume that all paraphrased questions (x, x') can be answered with a single, initially unknown, query (Table 4.1 shows example paraphrases). This assumption allows the algorithm to generalize from the initial seed lexicon L_0, greatly increasing the lexical coverage.

As an example, consider the paraphrase pair $x =$ "What is the population of New York?" and $x' =$ "How big is NYC?" Suppose x can be mapped to a query under L_0 using the following derivation y:

$$\begin{aligned} \text{What is the } \texttt{r} \text{ of } \texttt{e}? &= \texttt{(?, r, e)} \\ \text{population} &= \texttt{population} \\ \text{new york} &= \texttt{new york} \end{aligned}$$

We can induce new lexical items by aligning the patterns used in y to substrings in x'. For example, suppose we know that the words in (x, x') align in the following way:



Using this information, we can hypothesize that "How `r` is `e`," "big," and "NYC" should have the same interpretations as "What is the `r` of `e`," "population," and "New York," respectively, and create the new entries:

$$\begin{aligned} \text{How } \texttt{r} \text{ is } \texttt{e}? &= \texttt{(?, r, e)} \\ \text{big} &= \texttt{population} \\ \text{nyc} &= \texttt{new york} \end{aligned}$$

We call this procedure $\textsf{InduceLex}(x, x', y, A)$, which takes a paraphrase pair (x, x'), a derivation y of x, and a word alignment A, and returns a new set of lexical entries. Before formally describing InduceLex we need to introduce some definitions.

Let n and n' be the number of words in x and x'. Let $[k]$ denote the set of integers $\{1, \ldots, k\}$. A word alignment A between x and x' is a subset of $[n] \times [n']$. A phrase alignment is a pair of index sets (I, I') where $I \subseteq [n]$ and $I' \subseteq [n']$. A phrase alignment (I, I') is consistent with a word alignment A if for all $(i, i') \in A$, $i \in I$ if and only if $i' \in I'$. In other words, a phrase alignment is consistent with a word alignment if the words in the phrases are aligned only with each other, and not with any outside words.

We will now define $\textsf{InduceLex}(x, x', y, A)$ for the case where the derivation y consists of a 2-argument question entry (p_q, c_q), a relation entry (p_r, c_r), and an entity entry (p_e, c_e), as shown in the example above.[1] InduceLex returns the set of all triples $(p'_q, c_q), (p'_r, c_r), (p'_e, c_e)$ such that for all p'_q, p'_r, p'_e such that

1. p'_q, p'_r, p'_e are a partition of the words in x'.

2. The phrase pairs $(p_q, p'_q), (p_r, p'_r), (p_e, p'_e)$ are consistent with the word alignment A.

[1] InduceLex has similar behavior for the other type of derivation, which consists of a 1-argument question entry (p_q, c_q) and an entity (p_e, c_e).

3. The p'_r and p'_e are contiguous spans of words in x'.

Figure 4.1 shows the complete lexical learning algorithm. In practice, for a given paraphrase pair (x, x') and alignment A, InduceLex will generate multiple sets of new lexical entries, resulting in a lexicon with millions of entries. We use an existing statistical word alignment algorithm for WordAlign (see Section 4.4). In the next section, we will introduce a scalable approach for learning to score derivations to filter out lexical items that generalize poorly.

4.3.2 Weight Learning

Weight learning is necessary for filtering out derivations that use incorrect lexical entries like "new mexico" = `mexico`, which arise from noise in the paraphrases and noise in the word alignment. We use the hidden variable structured perceptron algorithm to learn $\mathbf{w}$ from a list of (question x, query z) training examples. We adopt the iterative parameter mixing variation of the perceptron (McDonald et al., 2010) to scale to a large number of training examples.

Figure 4.2 shows the weight learning algorithm. The weight learning algorithm operates in two stages. First, we use the initial lexicon L_0 to automatically generate (question x, query z) training examples from the paraphrase corpus $\mathcal{C}$. Then we feed the training examples into the learning algorithm, which estimates weights for the learned lexicon L.

Because the number of training examples is large, we adopt a parallel perceptron approach. We first randomly partition the training data $\mathcal{T}$ into K equally-sized subsets $\mathcal{T}_1, \ldots, \mathcal{T}_K$. We then perform perceptron learning on each partition in parallel. Finally, the learned weights from each parallel run are aggregated by taking a uniformly weighted average of each partition's weight vector. This procedure is repeated for T iterations.

The training data consists of (question x, query z) pairs, but our scoring model is over (question x, derivation y) pairs, which are unobserved in the training data. We use a hidden variable version of the perceptron algorithm (Collins, 2002), where the model weights are updated using the highest scoring derivation y^* that will generate the correct query z using the learned lexicon L.

function LEARNLEXICON

Inputs:

- A corpus $\mathcal{C}$ of paraphrases (x, x'). (Table 4.1)
- An initial lexicon L_0 of (pattern, concept) pairs.
- A word alignment function $\mathsf{WordAlign}(x, x')$. (Section 4.4)
- Initial weights $\mathbf{w}_0$.
- A function $\mathsf{Derivs}(x, L)$ that derives queries from a question x using lexicon L. (Section 4.2)
- A function $\mathsf{InduceLex}(x, x', y, A)$ that induces new lexical items from the paraphrases (x, x') using their word alignment A and a derivation y of x. (Section 4.3.1)

Output: A learned lexicon L.

$L = \{\}$

for all $x, x' \in \mathcal{C}$ **do**

  if $\mathsf{Derivs}(x, L_0)$ is not empty **then**

    $A \leftarrow \mathsf{WordAlign}(x, x')$

    $y^* \leftarrow \arg\max_{y \in \mathsf{Derivs}(x, L_0)} \mathbf{w}_0 \cdot \mathbf{f}(x, y)$

    $L \leftarrow L \cup \mathsf{InduceLex}(x, x', y^*, A)$

return L

Figure 4.1: The PARALEX lexicon learning algorithm.

function LEARNWEIGHTS

Inputs:

- A corpus $\mathcal{C}$ of paraphrases (x, x'). (Table 4.1)
- An initial lexicon L_0 of (pattern, KB concept) pairs.
- A learned lexicon L of (pattern, KB concept) pairs.
- Initial weights $\mathbf{w}_0$.
- Number of perceptron epochs T.
- Number of training-data shards K.
- A function $\mathsf{Derivs}(x, L)$ that derives queries from a question x using lexicon L. (Section 4.2)
- A function PerceptronEpoch$(\mathcal{T}, \mathbf{w}, L)$ that runs a single epoch of the hidden-variable structured perceptron algorithm on training set $\mathcal{T}$ with initial weights $\mathbf{w}$, returning a new weight vector $\mathbf{w}'$. (Section 4.3.2)

Output: A learned weight vector $\mathbf{w}$.

// Step 1: Generate Training Examples $\mathcal{T}$

$\mathcal{T} = \{\}$

for all $x, x' \in \mathcal{C}$ **do**

  if $\mathsf{Derivs}(x, L_0)$ is not empty **then**

    $y^* \leftarrow \arg\max_{y \in \mathsf{Derivs}(x, L_0)} \mathbf{w}_0 \cdot \mathbf{f}(x, y)$

    $z^* \leftarrow$ query of y^*

    Add (x', z^*) to $\mathcal{T}$

// Step 2: Learn Weights from $\mathcal{T}$

Randomly partition $\mathcal{T}$ into shards $\mathcal{T}_1, \ldots, \mathcal{T}_K$

for $t = 1 \ldots T$ **do**

  // Executed on k processors

  $\mathbf{w}_{k,t} = \text{PerceptronEpoch}(\mathcal{T}_k, \mathbf{w}_{t-1}, L)$

  // Average the weights

  $\mathbf{w}_t = \frac{1}{K} \sum_k \mathbf{w}_{k,t}$

return $\mathbf{w}_T$

Figure 4.2: The PARALEX weight learning algorithm.

4.4 Data

For our knowledge base K, we use the publicly available set of 15 million ReVerb extractions.[2] The knowledge base consists of a set of triples over a vocabulary of approximately 600K relations and 2M entities, extracted from the ClueWeb09 corpus.[3] The ReVerb knowledge base contains a large cross-section of general world-knowledge, and thus is a good testbed for Open QA. However, the extractions are noisy, unnormalized (*e.g.*, the entities `obama`, `barack obama`, and `president obama` all appear as distinct entities), and ambiguous (*e.g.*, the relation phrase `born in` contains facts about both dates and locations).

Our paraphrase corpus $\mathcal{C}$ was constructed from the collaboratively edited QA site WikiAnswers. WikiAnswers users can tag pairs of questions as alternate wordings of each other. We harvested a set of 18M of these question-paraphrase pairs, with 2.4M distinct questions in the corpus.

To estimate the precision of the paraphrase corpus, we randomly sampled a set of 100 pairs and manually tagged them as "paraphrase" or "not-paraphrase." We found that 55% of the sampled pairs are valid paraphrased. Most of the incorrect paraphrases were questions that were related, but not paraphrased *e.g.* "How big is the biggest mall?" and "Most expensive mall in the world?"

We word-aligned each paraphrase pair using the MGIZA++ implementation of IBM Model 4 (Och and Ney, 2000; Gao and Vogel, 2008). The word-alignment algorithm was run in each direction (x, x') and (x', x) and then combined using the `grow-diag-final-and` heuristic (Koehn et al., 2003).

4.5 Experimental Setup

We compare the following systems:

- Paralex: the full system, using the lexical learning and weight learning algorithms from Section 4.3.

[2] We used version 1.1, downloaded from `http://reverb.cs.washington.edu/`.

[3] The full set of ReVerb extractions from ClueWeb09 contains billions of triples. We used the smaller subset of triples to simplify the Paralex experiments. Chapter 5 explores Open QA over a larger set of extractions.

- **NoWeight:** PARALEX without the learned weights.
- **InitOnly:** PARALEX using only the initial seed lexicon.

We evaluate the systems' performance on the end-task of QA on WikiAnswers questions.

4.5.1 Test Set

Our goal is to measure PARALEX's performance on question interpretation. A major challenge for evaluation is that the REVERB knowledge base is incomplete. A system may correctly map a test question to a valid query, only to return no results when executed against the incomplete knowledge base. We factor out this source of error by semi-automatically constructing a sample of questions that are known to be answerable using the REVERB knowledge base. This allows us to measure the performance of the systems on question interpretation.

To create the evaluation set, we identified questions x in a held out portion of the WikiAnswers corpus such that (1) x can be mapped to some query z using an initial lexicon (described in Section 4.5.4), and (2) when z is executed against the knowledge base, it returns at least one answer. We then add x and all of its paraphrases as our evaluation set. For example, the question "What is the language of Hong-Kong?" satisfies these requirements, so we added these questions to the evaluation set:

What is the language of Hong-Kong?
What language do people in hong kong use?
How many languages are spoken in hong kong?
How many languages hong kong people use?
In Hong Kong what language is spoken?
Language of Hong-kong?

This methodology allows us to evaluate the systems' ability to handle syntactic and lexical variations of questions that should have the same answers. We created 37 question clusters, resulting in a total of 698 questions. We removed all of these questions and their paraphrases from the training set. We also manually filtered out any incorrect paraphrases that appeared in the test clusters.

Question Pattern	Knowledge Base Query
Who `r` `e`?	`(?, r, e)`
What `r` `e`?	`(?, r, e)`
Who does `e` `r`?	`(e, r, ?)`
What does `e` `r`?	`(e, r, ?)`
What is the `r` of `e`?	`(?, r, e)`
Who is the `r` of `e`?	`(?, r, e)`
What is `r` by `e`?	`(e, r, ?)`
Who is `e`'s `r`?	`(?, r, e)`
What is `e`'s `r`?	`(?, r, e)`
Who is `r` by `e`?	`(e, r, ?)`
When did `e` `r`?	`(e, r in, ?)`
When did `e` `r`?	`(e, r on, ?)`
When was `e` `r`?	`(e, r in, ?)`
When was `e` `r`?	`(e, r on, ?)`
Where was `e` `r`?	`(e, r in, ?)`
Where did `e` `r`?	`(e, r in, ?)`

Table 4.3: The question patterns used in the initial lexicon L_0.

We then created a gold-standard set of (x, a, l) triples, where x is a question, a is an answer, and l is a label (correct or incorrect). To create the gold-standard, we first ran each system on the evaluation questions to generate (x, a) pairs. Then we manually tagged each pair with a label l. This resulted in a set of approximately $2,000$ human judgments. If (x, a) was tagged with label l and x' is a paraphrase of x, we automatically added the labeling (x', a, l), since questions in the same cluster should have the same answer sets. This process resulted in a gold standard set of approximately $48,000$ (x, a, l) triples.

4.5.2 Metrics

We compute precision and recall of each system's top answer. Precision is the fraction of predicted answers that are correct. Recall is the fraction of questions where a correct answer was predicted.

4.5.3 Features and Settings

The feature representation $\mathbf{f}(x, y)$ consists of indicator functions for each lexical entry $(p, c) \in L$ used in the derivation y. For weight learning, we use an initial weight vector $\mathbf{w}_0 = 0$, use $T = 20$ iterations and shard the training data into $K = 10$ pieces. We limit each system to return the top 100 knowledge base queries for each test sentence. All input words are lowercased and lemmatized.

4.5.4 Initial Lexicon

Both the lexical learning and weight learning algorithms rely on an initial seed lexicon L_0. The initial lexicon allows the learning algorithms to bootstrap from the paraphrase corpus.

We construct L_0 from a set of 16 hand-written, 2-argument question patterns and the output of the identity transformation on the entity and relation strings in the knowledge base. Table 4.3 shows the question patterns that were used in L_0.

4.6 Results

Figure 4.4 shows the performance of PARALEX on the test questions. PARALEX outperforms the baseline systems in terms of F1. The lexicon-learning algorithm boosts the recall by a factor of four over the initial lexicon, showing the utility of the PARALEX algorithm. The weight-learning algorithm also results in a large gain in both precision and recall: PARALEX generates a noisy set of patterns, so selecting the best query for a question is more challenging.

Figure 4.3 shows an ablation of the different types of lexical items learned by PARALEX. For each row, we removed the learned lexical items from each of the types described in Section 4.2, keeping only the initial seed lexical items. The learned 2-argument question

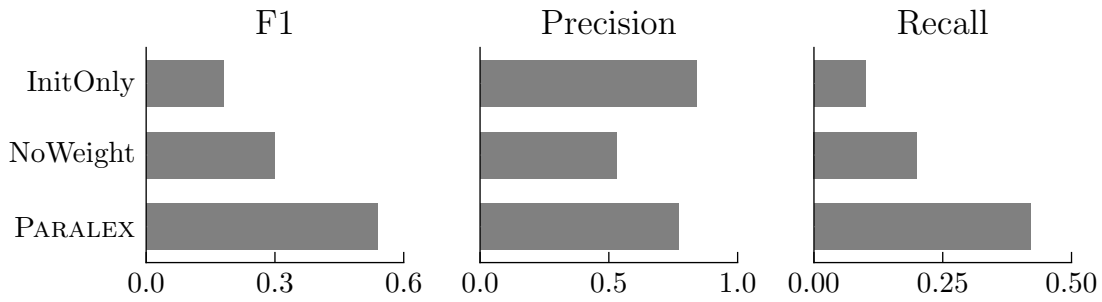


Table 4.4: Performance on WikiAnswers questions known to be answerable using ReVerb.

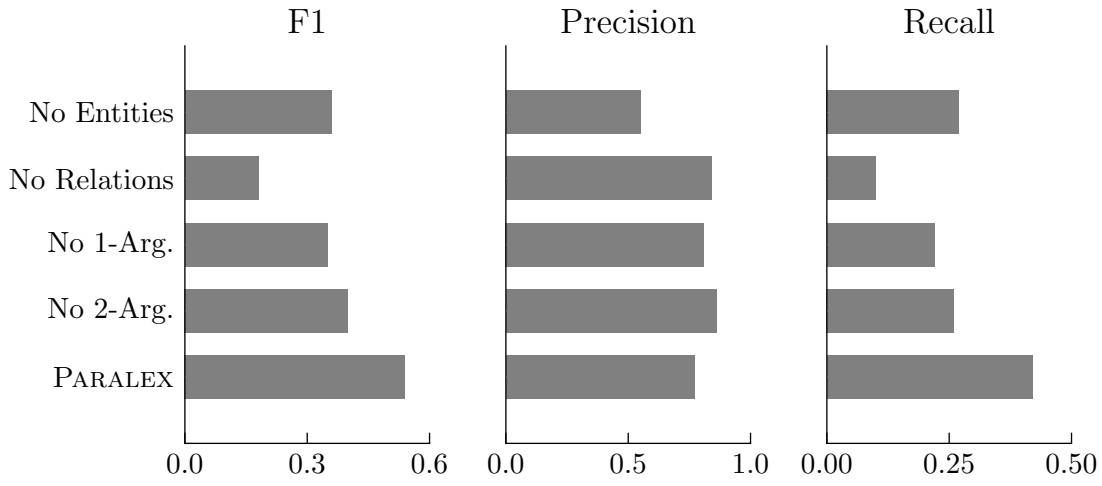


Figure 4.3: Ablation of the learned lexical items.

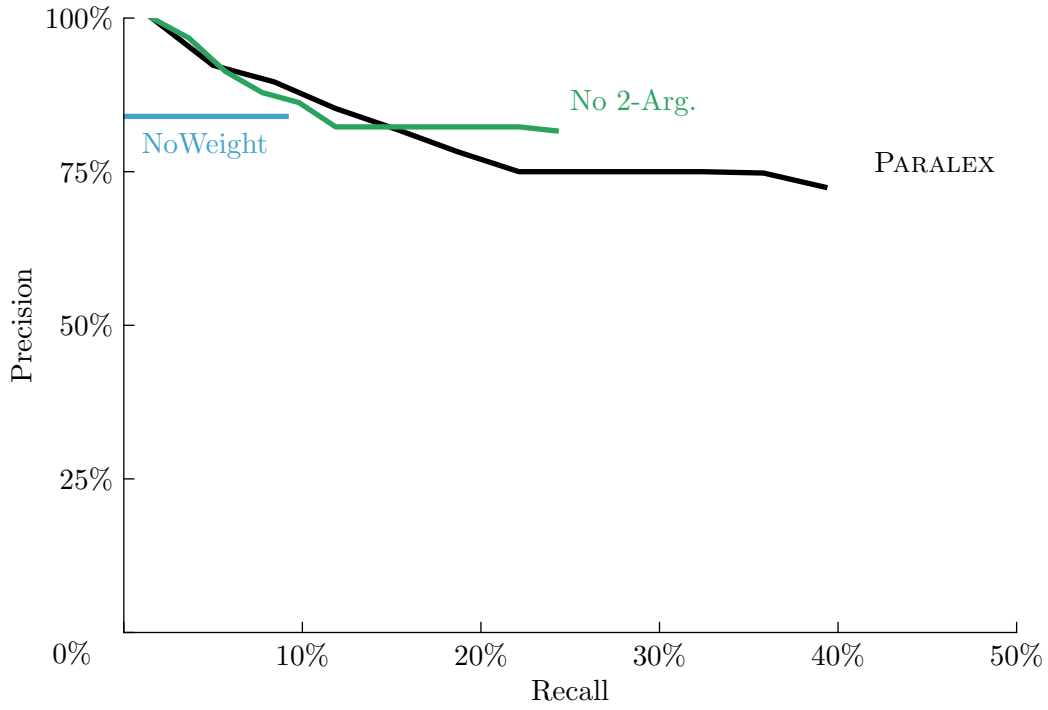


Figure 4.4: Precision-recall curves for PARALEX with and without 2-argument question patterns.

String	Learned Knowledge Base Relations for String
get rid of	`treatment for`, `cause`, `get rid of`, `cure for`, `easiest way to get rid of`
word	`word for`, `slang term for`, `definition of`, `meaning of`, `synonym of`
speak	`speak language in`, `language speak in`, `principal language of`, `dialect of`
useful	`main use of`, `purpose of`, `importance of`, `property of`, `usefulness of`

String	Learned Knowledge Base Entities for String
smoking	`smoking`, `tobacco smoking`, `cigarette`, `smoking cigar`, `smoke`, `quit smoking`
radiation	`radiation`, `electromagnetic radiation`, `nuclear radiation`
vancouver	`vancouver`, `vancouver city`, `vancouver island`, `vancouver british columbia`
protein	`protein`, `protein synthesis`, `plasma protein`, `monomer`, `dna`

Table 4.5: Examples of relation and entity synonyms learned from the WikiAnswers paraphrase corpus.

templates increase the recall of the system. This increased recall came at a cost, lowering precision from 0.86 to 0.77. Thresholding the query score allows us to trade precision for recall, as shown in Figure 4.4. Table 4.5 shows some examples of the learned entity and relation synonyms.

The 2-argument question templates help PARALEX generalize over different variations of the same question, like the test questions shown in Table 4.6. For each question, PARALEX combines a 2-argument question template (shown below the questions) with the rules "celebrate" = `holiday of` and "christians" = `christians` to derive a full query. Factoring the problem this way allows PARALEX to reuse the same rules in different syntactic configurations. Note that the imperfect training data can lead to overly-specific templates like "What are the religious `r` of `e`," which can lower accuracy.

Celebrations for Christians?
`r` for `e`?
Celebrations of Christians?
`r` of `e`?
What are some celebrations for Christians?
What are some `r` for `e`?
What are some celebrations of the Christians?
What are some `r` of `e`?
What are some of Christians celebrations?
What are some of `e` `r`?
What celebrations do Christians do?
What `r` do `e` do?
What did Christians celebrate?
What did `e` `r`?
What are the religious celebrations of Christians?
What are the religious `r` of `e`?
What celebration do Christians celebrate?
What `r` do `e` celebrate?

Table 4.6: Questions from the test set with 2-argument question patterns that PARALEX used to derive a correct query.

4.7 Error Analysis

To understand how close we are to the goal of Open QA, including both knowledge acquisition and question interpretation, we ran PARALEX on an unrestricted sample of questions from WikiAnswers. We used the same methodology as described in the previous section, where PARALEX returns the top answer for each question using REVERB.

We found that PARALEX performs worse on this dataset, with recall maxing out at approximately 6% of the questions answered at precision 0.4. This is not surprising, since the test questions are not restricted to topics covered by the REVERB knowledge base, and may be too complex to be answered by any knowledge base of relational triples.

We performed an error analysis on a sample of 100 questions that were either incorrectly answered or unanswered. We examined the candidate queries that PARALEX generated for each question and tagged each query as correct (would return a valid answer given a correct and complete knowledge base) or incorrect. Because the input questions are unrestricted, we also judged whether the questions could be faithfully represented as a $(?, r, e)$ or $(e, r, ?)$ query over the knowledge base vocabulary. Table 4.7 shows the distribution of errors.

The largest source of error (36%) were on questions that could not be represented as a single-relation query. We categorized these questions into groups. The largest group (14%) were questions that need n-ary or higher-order knowledge base relations, for example "How long does it take to drive from Sacramento to Cancun?" or "What do cats and dogs have in common?" Approximately 13% of the questions were how-to questions like "How do you make axes in minecraft?" whose answers are a sequence of steps, instead of a knowledge base entity. Lastly, 9% of the questions require knowledge base operators like joins, for example "When were Bobby Orr's children born?"

The second largest source of error (32%) were questions that could be represented as a query, but where PARALEX was unable to derive any correct queries. For example, the question "Things grown on Nigerian farms?" was not mapped to any queries, even though the REVERB knowledge base contains the relation `grown in` and the entity `nigeria`. We found that 13% of the incorrect questions were cases where the entity was not recognized, 12% were cases where the relation was not recognized, and 6% were cases where both the

entity and relation were not recognized.

We found that 28% of the errors were cases where PARALEX derived a query that we judged to be correct, but returned no answers when executed against the knowledge base. For example, given the question "How much can a dietician earn?" PARALEX derived the query `(?, salary of, dietician)` but this returned no answers in the REVERB knowledge base.

Finally, approximately 4% of the questions included typos or were judged to be inscrutable, for example "Barovier hiriacy of evidence based for pressure sore?"

Discussion Our experiments show that the learning algorithms described in Section 4.3 allow PARALEX to generalize beyond an initial lexicon and answer questions with higher accuracy. Our error analysis on an unrestricted set of WikiAnswers questions shows that PARALEX is still far from the goal of high-recall Open QA. We found that many questions asked on WikiAnswers are either too complex to be mapped to a simple relational query, or are not covered by the REVERB knowledge base. Further, approximately one third of the missing recall is due to question interpretation errors.

4.8 Conclusion

We introduced a new learning approach that induces a complete Open QA system from a large corpus of noisy question paraphrases. Using only a seed lexicon, the approach automatically learns a lexicon and linear ranking function that demonstrated high accuracy on a held-out evaluation set. In the next Chapter, we introduce the OQA system, which leverages both curated and extracted knowledge bases for Open QA and outperforms PARALEX in terms of precision and recall.

Incorrectly Answered/Unanswered Questions	
36%	Complex Questions
	Need n-ary or higher-order relations (14%)
	Answer is a set of instructions (13%)
	Need operators *e.g.* joins (9%)
32%	Entity or Relation Recognition Errors
	Entity recognition errors (13%)
	Relation recognition errors (12%)
	Entity & relation recognition errors (7%)
28%	Incomplete Knowledge Base
	Derived a correct query, but no answers
4%	Typos/Inscrutable Questions

Table 4.7: Error distribution of PARALEX on an unrestricted sample of questions from the WikiAnswers dataset.

Chapter 5

OPEN QUESTION ANSWERING OVER CURATED AND EXTRACTED KNOWLEDGE BASES

In the previous chapter, I introduced the PARALEX system, which was the first to perform Open QA over an extracted knowledge base. Recently, there has been work on scaling semantic parsing systems to work over curated knowledge bases like Freebase (Cai and Yates, 2013; Kwiatkowski et al., 2013; Berant et al., 2013). In this chapter, I introduce OQA, the first Open QA system to use both curated and extracted knowledge bases to derive answers.

A key challenge in Open QA is to be robust to the high variability found in natural language and the many ways of expressing knowledge in large-scale KBs. OQA achieves this robustness by decomposing the full QA problem into smaller sub-problems that are easier to solve. Figure 5.1 shows an example of how OQA maps the question "How can you tell if you have the flu?" to the answer "chills" over four steps. The first step rewrites the input question to "What are signs of the flu?" using a paraphrase operator mined from a large corpus of questions. The second step uses a hand-written template to parse the paraphrased question to the KB query `(?x, signs of, the flu)`. These two steps are synergistic; paraphrase operators effectively reduce the variance of the input questions, allowing OQA to use a small set of high-precision parsing rules while maintaining recall. The third step uses a query-rewrite operator to reformulate the query as `(the flu, symptoms, ?x)`. Query-rewrite operators are automatically mined from the KB, and allow the vocabulary mismatch between question words and KB symbols to be solved independent of parsing. Finally, the fourth step executes the rewritten query against the KB, returning the final answer.

The operators and KB are noisy, so it is possible to construct many different sequences of operations (called derivations), very few of which will produce a correct answer. OQA learns from a small amount of question-answer data to find the best derivations. Because

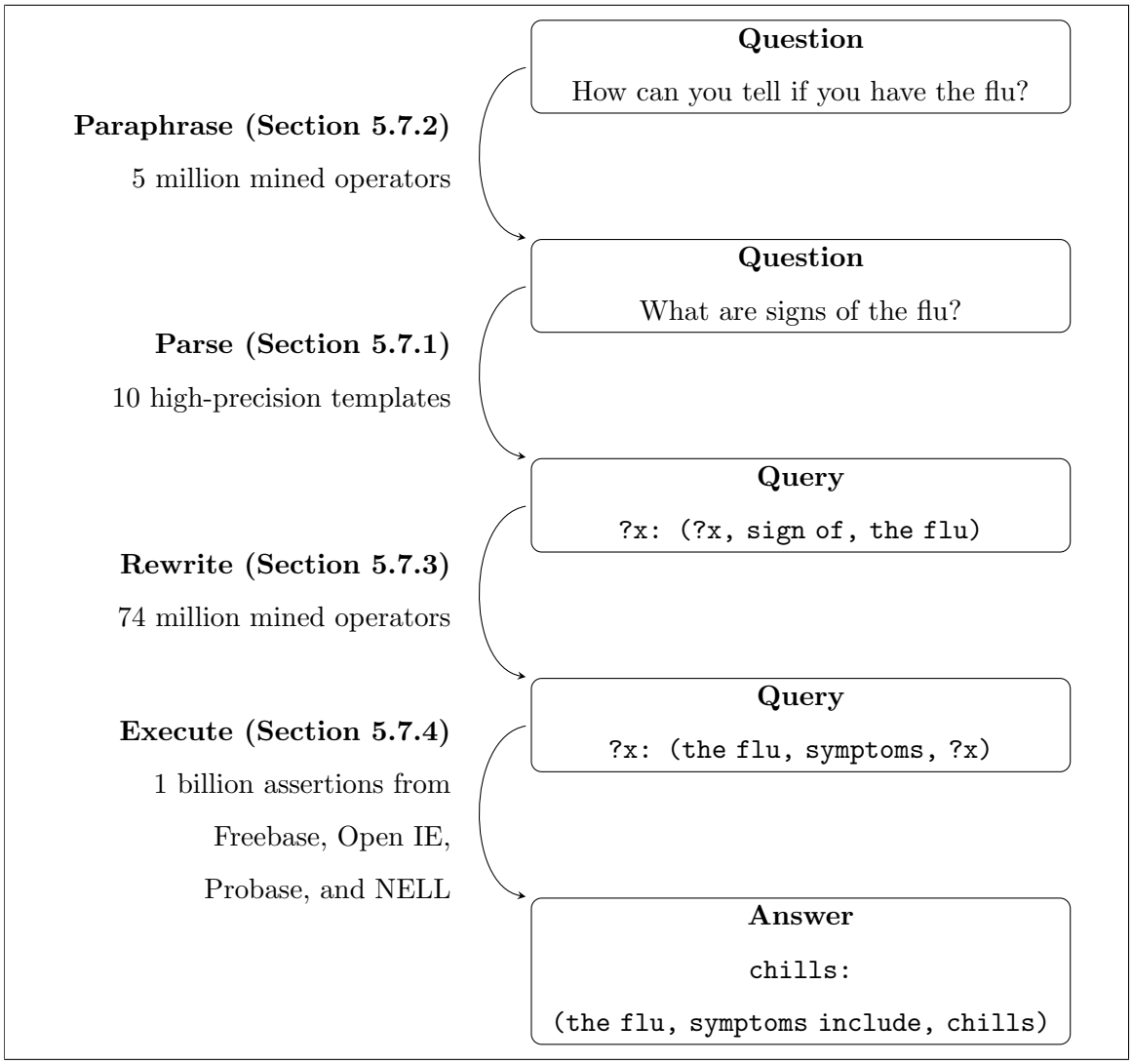


Figure 5.1: OQA automatically mines millions of operators (left) then learns to compose them to answer questions (right) using evidence from multiple knowledge bases.

the derivations are unobserved in the training data, we use a latent-variable structured perceptron algorithm (Zettlemoyer and Collins, 2005; Liang et al., 2006; Sun et al., 2009). OQA uses a small set of general features that allow it to generalize from a limited number of training examples. Experiments on three benchmark question sets show that OQA outperforms PARALEX, achieving twice the precision and recall.

In summary, we make the following contributions:

- We introduce OQA, the first Open QA system to leverage multiple, large-scale curated and extracted KBs.

- We describe an inference algorithm for deriving high-confidence answers (Section 5.5) and a hidden-variable structured perceptron algorithm for learning a scoring function from data (Section 5.6).

- We present algorithms for automatically mining paraphrase operators from a question corpus (Section 5.7.2) and KB-query rewrite operators (Section 5.7.3) from multiple KBs.

- We provide an empirical evaluation (Section 5.8), showing the relative contributions of different KBs and different system components across three question sets. We also compare OQA to the state-of-the-art QA systems PARALEX (Fader et al., 2013) and SEMPRE (Berant et al., 2013).

In Section 5.1, we describe related work in more detail before moving on to the description of OQA (Sections 5.2–5.7) and experiments (Section 5.8).

5.1 Related Work

PARALEX was the first Open QA system to operate over a noisy, extracted KB. The biggest difference between PARALEX and OQA is how they decompose the QA problem. PARALEX uses self-labeled data to learn templates that directly map questions to queries—essentially performing paraphrasing, parsing, and query-rewriting in one step. OQA treats these as

separate problems, which allows it to combine high-recall data mining techniques (for paraphrasing and query rewriting) with high-precision, hand-written rules (for parsing).

OQA's feature representation also differs from previous work. Previous systems use a large number of lexicalized features—those involving specific lexemes or KB symbols. OQA uses unlexicalized features that operate on the level of function words, part-of-speech tags, and corpus statistics. We found that the an unlexicalized feature representation generalizes better to questions involving relationships that were never seen during training.

5.2 Task Definition and Overview

In this section, we define the QA task and give a high-level outline of OQA and our experiments.

Task and Metrics: We focus on the task of factoid QA, where the system takes a natural language question like "How can you tell if you have the flu?" as input and returns a short string answer like `chills` from a KB, or "no answer." We use precision (fraction of answered questions that are correct) and recall (fraction of questions that are correctly answered) as our primary evaluation metrics.

Knowledge Base: OQA uses a simple KB abstraction where ground facts are represented as string triples (argument1, relation, argument2). We use triples from curated and extracted knowledge sources (Section 5.3.1) and provide a lightweight query language to access the KB (Section 5.3.2).

Operators and Scoring Function: OQA models QA as a process where answers are derived from questions using operators Ops (Section 5.4.1). A sequence of operators linking a question to an answer is called a derivation. OQA computes the confidence of an answer derivation d using a linear scoring function $\text{score}(d|\mathbf{f}, \mathbf{w})$, which is parameterized by a feature function $\mathbf{f}$ and feature weights $\mathbf{w}$ (Section 5.4.2).

Inference: In practice, the space of derivations defined by Ops is too large to enumerate. OQA uses heuristic search over partial derivations, guided by $\text{score}(d|\mathbf{f}, \mathbf{w})$, to generate high-scoring candidate answers for an input question (Section 5.5).

Source	Type	# Triples	# Relation Phrases
Freebase	Curated	300M	18K
Open IE (Fader et al., 2011)	Extracted	500M	6M
Probase (Wu et al., 2012)	Extracted	200M	1
NELL (Carlson et al., 2010)	Extracted	2M	300

Table 5.1: Knowledge bases used by OQA.

Learning: OQA learns the weights **w** from a small set of question-answer pairs. Because annotated answer derivations are difficult to obtain, we use a latent-variable structured perceptron algorithm that treats answer derivations as unobserved variables in the training data (Section 5.6).

Operators and Features: OQA uses four types of operators: a small set of parsing operators (Section 5.7.1), a large set of paraphrase operators mined from a question corpus (Section 5.7.2), a large set of query-rewrite rules mined from the KB (Section 5.7.3), and an execution operator that interfaces with the KB (Section 5.7.4). Each operator is paired with a small set of features used to compute **f**. Using a small feature set results in better generalization from limited training data.

System Evaluation: We evaluate OQA using three question sets. We compare OQA to the Open QA system PARALEX and the Freebase QA system SEMPRE (Berant et al., 2013). We then test the contributions of each knowledge source and system component via ablation.

5.3 Knowledge Base

This section describes where OQA's knowledge comes from and the query language it uses to access the knowledge.

5.3.1 Knowledge Base Sources

Table 5.1 summarizes the knowledge sources in OQA. OQA uses one curated KB (Freebase) and three extracted KBs (Open IE, Probase, and NELL).

Freebase is an open-domain, collaboratively edited KB. Freebase has comprehensive coverage of certain domains like film or geography, but does not contain imprecise assertions like "chicken is high in protein." Freebase maintains canonical string representations of its entities and relations, which we use to coerce facts into string triples.[1]

Open IE (Banko et al., 2007; Fader et al., 2011) is a family of techniques used to extract binary relationships from billions of web pages containing unstructured text. Open IE has the unique property that its relations are unnormalized natural language, which results in over two orders of magnitude more relation phrases than Freebase. The Open IE assertions are noisy and lack the comprehensive domain coverage found in Freebase. However, Open IE contains many of the informal assertions that are not found in curated KBs. For example, Open IE produces assertions like `(pepper, provides a source of, vitamins a and c)`. Open IE triples are annotated with metadata including extractor confidence and corpus frequency, and some triple arguments are linked to Freebase entities (Lin et al., 2012).

Probase (Wu et al., 2012) is an extracted KB containing "is-a" relations, *e.g.*, `(paris, is-a, beautiful city)` or `(physicist, is-a, scientist)`. Probase triples are annotated with statistical metadata that measure the confidence of each extraction.

NELL (Carlson et al., 2010) is an extracted KB that contains approximately 300 relation phrases. NELL generally has high precision, but low recall.

The union of these KBs forms a single resource containing a billion noisy, redundant, and inconsistent assertions. While there is a vast body of literature exploring the problem of data integration (Doan et al., 2012), these techniques require a target schema, which does not exist for our knowledge sources. Instead of making an offline commitment to a single schema, at runtime OQA hypothesizes many interpretations for each question. These hypotheses are encoded using the query language described in the next section.

[1] We discard Freebase facts that are not binary relations.

What fruits are a source of vitamin C?

```
?x :  (?x, is-a, fruit) (?x, source of, vitamin c)
```

```
SELECT t0.arg1 FROM triples AS t0, triples AS t1
  WHERE
    keyword-match(t0.rel, "is-a")        AND
    keyword-match(t0.arg2, "fruit")      AND
    keyword-match(t1.rel, "source of")   AND
    keyword-match(t1.arg2, "vitamin c")  AND
    string-similarity(t0.arg1, t1.arg1) > 0.9
```

t0.arg1	t0.rel	t0.arg2	t1.arg1	t1.rel	t1.arg2
Lychee	is a	fruit	Lychees	good source of	vitamin c
star-fruit	is a	tropical fruit	starfruit	source of	vitamin c
pepper	is a	fresh fruit	pepper	provides a source of	vitamins c and a

Figure 5.2: Top: An example question and query used by OQA. Middle: The query semantics expressed as SQL. Bottom: The results when executed against a knowledge base (answers highlighted).

5.3.2 Query Language

The query language used in OQA provides a lightweight interface between natural language and KB assertions. In contrast to the semantic parsing literature (Zelle and Mooney, 1996), a OQA query is not intended to represent the complete, formal semantic interpretation of a question. Instead, the query language is used to separate the parsing problem (identifying predicate-argument structure) from the vocabulary-matching problem (matching natural language symbols to KB symbols) (Grosz et al., 1987; Kwiatkowski et al., 2013). This factorization is at the core of the OQA approach, which uses different operators to solve

each problem.

OQA's query language is capable of representing conjunctive queries (Chandra and Merlin, 1977). Because our KB is unnormalized and contains only strings, OQA uses keyword matching and string similarity as primitive operations. Figure 5.2 shows how the question "What fruits are a source of vitamin C?" can be represented as the query `?x : (?x, is-a, fruit) (?x, source of, vitamin c)`. This particular query represents one possible mapping of the question to a predicate-argument structure. The middle box of Figure 5.2 shows how the semantics of the query can be interpreted as a SQL expression over a single table `triples` with string columns `arg1`, `rel`, and `arg2`. A OQA query consists of a projection variable (*e.g.*, `?x`) and a list of conjuncts. Each conjunct contains a mix of string literals (*e.g.*, `fruit`) and variables. String literals correspond to keyword-matching constraints on the table columns, while variables correspond to string-similarity join constraints.

Having keyword matching and string similarity incorporated into the query semantics leads to another useful factorization. The query language provides a general, high-recall solution to the problem of minor surface-form variations (*e.g.*, joining `star-fruit` with `starfruit` or matching `source of` with `provides a source of` in Figure 5.2). OQA can then increase precision by computing question- or KB-specific features as soft constraints on the output. For example, it uses a feature that checks whether two join keys are linked to the same Freebase entity, if this information is available. This lets OQA maintain a simple data model (entity-linking is not required) while allowing for domain knowledge to be modeled via features.

5.4 Deriving and Scoring Answers

OQA factors question answering into a set of smaller, related problems including paraphrasing, parsing, and query reformulation. The solutions to each of these sub-problems can then be applied in sequence to give a complete mapping from question to answer. Figure 5.3 shows example derivations for the question "How can you tell if you have the flu?" Our approach consists of two parts: (1) derivation operators, which define the space of possible answers for a given question, and (2) a scoring function, which returns a real-valued confidence for a derivation.

5.4.1 Derivation Operators

More formally, we model question answering as the process of mapping a question q to an answer a by applying operators from some set Ops. Each operator $o \in \mathsf{Ops}$ takes a state object $s \in \mathsf{States}$ as input and returns a set $o(s) \subseteq \mathsf{States}$ of successor states as output. State objects encode intermediate values that are used during the question-answering procedure. In Figure 5.3, the intermediate question "What are signs of the flu?" is a state object that is related to the query `?x : (?x, sign of, flu)` via a parsing operator. We use three types of states: question states, query states, and answer states.

Operations can be chained together into a derivation. A single derivation step encodes the process of applying an operator o to some state s and picking a successor state $s' \in o(s)$ from the output. A derivation $d = (\mathbf{o}, \mathbf{s}, k)$ consists of a sequence of k operators $\mathbf{o} = (o_1, \ldots, o_k)$ and a sequence of $k+1$ states $\mathbf{s} = (s_0, s_1, \ldots, s_k)$ satisfying $s_i \in o_i(s_{i-1})$ for all $1 \leq i \leq k$.[2]

An answer a is derivable from a question q under the operator set Ops if there exists some derivation $(\mathbf{o}, \mathbf{s}, k)$ such that $s_0 = q$ and $s_k = a$. We use the notation $\mathsf{Derivs}(q, \mathsf{Ops})$ to represent the space of all possible derivations from the question q under the operations Ops ending at answer a.

In our implementation of OQA, the operator set Ops contains millions of operators, combining both hand-written operators and operators learned from data. These operators are noisy: incorrect answers can be derived from most questions. Thus, estimating the confidence of a derivation is necessary for returning answers with high precision.

5.4.2 Scoring Function

To compute the confidence of a derivation, OQA uses a scoring function. The scoring function computes a real value for a given derivation, where large, positive scores are assigned to high-confidence derivations.

We make two assumptions about the form of the scoring function. First, we assume that

[2] In OQA, $2 \leq k \leq 4$: parsing and execution steps are required to derive an answer, and we limit derivations to have at most one paraphrase step and one query-rewrite step.

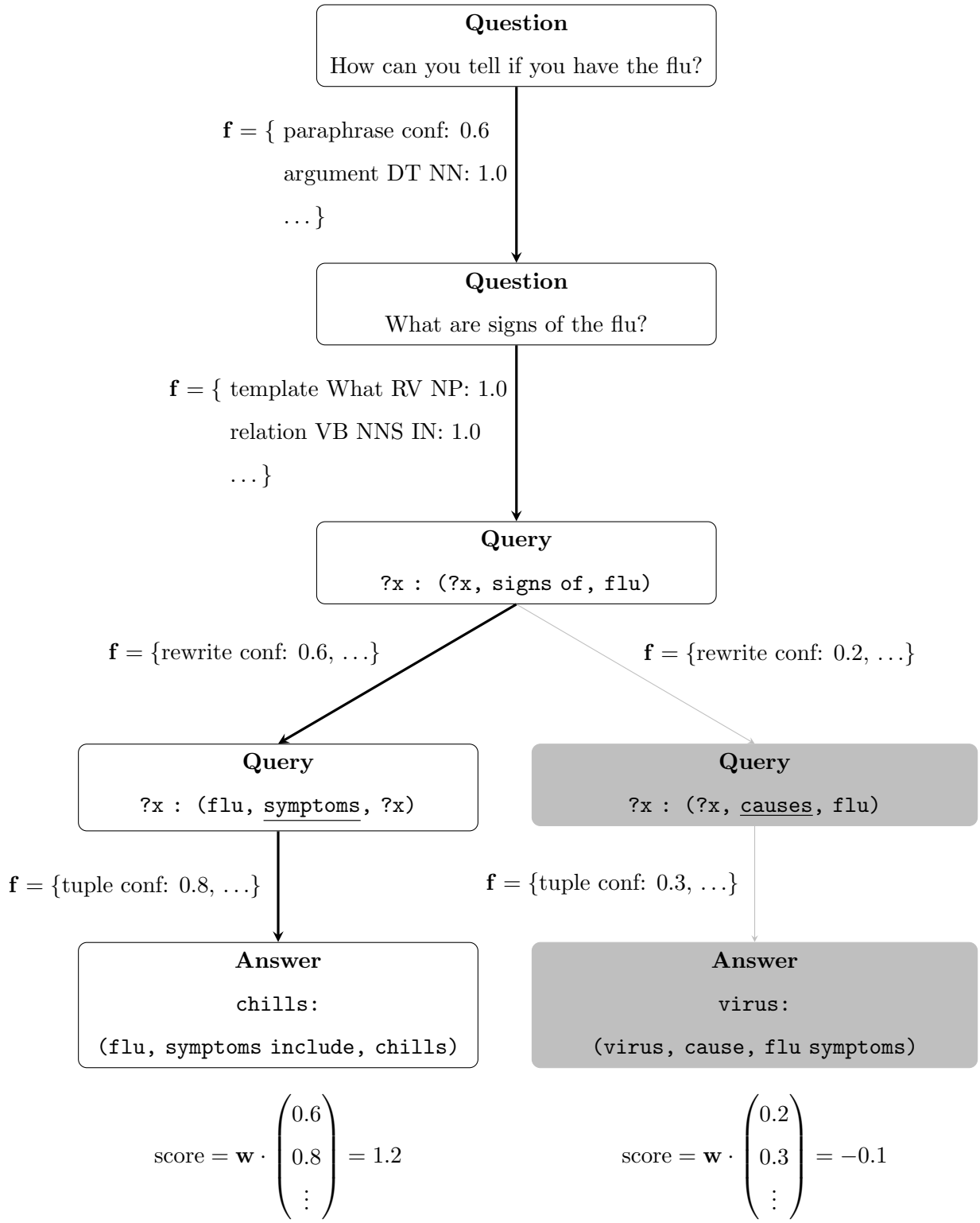


Figure 5.3: OQA computes operator-specific features to discriminate between correct derivations (left path) and incorrect derivations (right path).

the score is a linear function over features computed from a derivation. This will allow us to use familiar algorithms to learn the function from data. Second, we assume that the feature function decomposes over derivation steps. This allows us to use the scoring function to score partial derivations, which is useful for searching over $\mathsf{Derivs}(q, \mathsf{Ops})$ (discussed further in Section 5.5).

Under these assumptions, the score of a derivation $d = (\mathbf{o}, \mathbf{s}, k)$ can be written as

$$\text{score}(d|\mathbf{f}, \mathbf{w}) = \sum_{i=1}^{k} \mathbf{w} \cdot \mathbf{f}(s_0, s_{i-1}, o_i, s_i), \tag{5.1}$$

where $\mathbf{f}$ is an n-dimensional feature function that maps a derivation step into $\mathbb{R}^n$ and $\mathbf{w}$ is an n-dimensional weight vector. Because s_0 (the input question) is a constant in all derivations, we pass it as an argument to the feature function. This allows features to compute properties relating derivation steps to the input question.

Figure 5.3 shows an example of how the scoring function can discriminate between a correct answer (left) and an incorrect answer (right). The derivation on the left rewrites the original query using a high-confidence rule and uses high-confidence evidence returned from the KB. The derivation on the right uses a low-confidence rewrite rule and low-confidence evidence—and is thus assigned a lower score.

In our implementation of OQA, we learn $\mathbf{w}$ from a small set of question-answer pairs (Section 5.6) and define $\mathbf{f}$ to compute operator-specific features (Section 5.7).

5.5 Inference

We focus on the task of finding a single answer with the highest confidence under the scoring function, which amounts to solving the following equation for an input question q:

$$d^* = \underset{d \in \mathsf{Derivs}(q, \mathsf{Ops})}{\arg\max} \text{score}(d|\mathbf{f}, \mathbf{w}). \tag{5.2}$$

The underlying knowledge base and set of operators are both large enough that exhaustively enumerating $\mathsf{Derivs}(q, \mathsf{Ops})$ is not feasible. Instead, OQA uses beam search informed by the scoring function to explore $\mathsf{Derivs}(q, \mathsf{Ops})$. We refer to the beam search routine as DeriveAnswers.

The algorithm takes a question q as input and returns a set of derivations $D \subseteq \mathsf{Derivs}(q, \mathsf{Ops})$. The output set D is constructed by iteratively growing partial derivations starting at the initial question state $s_0 = q$. The algorithm maintains a beam of partial derivations, each scored by the function $\text{score}(d|\mathbf{f}, \mathbf{w})$. At every iteration, a partial derivation is selected to be extended. Extending a derivation $d = (\mathbf{o}, \mathbf{s}, k)$ amounts to computing successors to the state s_k and appending the successors to construct new derivations. This process is repeated until there are no partial derivations left to extend or until a time limit is reached.

In practice, the scoring function $\text{score}(d|\mathbf{f}, \mathbf{w})$ generally assigns higher scores to short derivations, *e.g.* derivations that do not use a query-rewrite operator. We found that this bias will flood the beam with high-scoring partial derivations occurring early in the search, and later options will not be considered. To avoid this problem, we maintain separate beams for each state-type in the search, similar to the decoding algorithms used in statistical machine translation (Koehn, 2004). OQA uses beam search both at runtime and during learning, which we describe in the next section.

5.6 Learning

A key challenge in learning $\text{score}(d|\mathbf{f}, \mathbf{w})$ is that obtaining labeled answer derivations requires expert annotators and is time consuming. Following recent work in semantic parsing (Clarke et al., 2010; Berant et al., 2013; Kwiatkowski et al., 2013), we use question-answer pairs as indirect supervision and treat answer derivations as unobserved variables in the training data. Question-answer pairs like $q =$ "How can you tell if you have the flu?", $A = \{$`chills`, `fever`, `aches`$\}$ are easier to obtain and do not require expert annotators.

We use the latent-variable structured perceptron algorithm (Zettlemoyer and Collins, 2005; Liang et al., 2006; Sun et al., 2009) to learn $\mathbf{w}$ from example question-answer pairs. Figure 5.4 shows the pseudocode for the LEARNWEIGHTS algorithm.

The algorithm takes as input a set of N pairs (q_i, A_i) for $i = 1, \ldots, N$, where A_i is a set containing string answers to q_i. For each training example, the algorithm calls DERIVEANSWERS to generate a candidate set of answer derivations D. The algorithm then chooses a derivation $\hat{d}$ that has the highest score according to the current weights and makes the prediction $\hat{a} = \text{answer}(\hat{d})$. If $\hat{a}$ is correct (*i.e.*, it is in A_i), the algorithm proceeds to the next

function LEARNWEIGHTS

Inputs:

Number of iterations T

N example questions with answers $(q_1, A_1), \ldots, (q_N, A_N)$

Initial model $(\mathsf{Ops}, \mathbf{f}, \mathbf{w})$ (Defined in Section 5.4)

Function DERIVEANSWERS (Defined in Section 5.5)

Output:

Learned weights $\mathbf{w}$

for $t = 1, \ldots, T$ **do**

  for $i = 1, \ldots, N$ **do**

    $D = \text{DERIVEANSWERS}(q_i, \mathsf{Ops}, \mathbf{f}, \mathbf{w})$

    $\hat{d} = \arg\max_{d \in D} \text{score}(d|\mathbf{f}, \mathbf{w})$

    if $\text{answer}(\hat{d}) \notin A_i$ **then**

      $D^* = \{d \in D : \text{answer}(d) \in A_i\}$

      $d^* = \arg\max_{d \in D^*} \text{score}(d|\mathbf{f}, \mathbf{w})$

      $\mathbf{w} = \mathbf{w} + \mathbf{f}(d^*) - \mathbf{f}(\hat{d})$

return average of $\mathbf{w}$ over all iterations

Figure 5.4: The weight-learning algorithm.

example. If $\hat{a}$ is incorrect, then the learner picks the highest scoring derivation d^* such that $\text{answer}(d^*)$ is in A_i. The algorithm then performs an additive update $\mathbf{w} = \mathbf{w} + \mathbf{f}(d^*) - \mathbf{f}(\hat{d})$. If there are no derivations with a correct answer in D, then the learner immediately proceeds to the next example without performing an update. Finally, the algorithm returns the average value of $\mathbf{w}$ over all iterations, which improves generalization (Freund and Schapire, 1999).

5.7 Operators and Features

In this section, we describe the operators that OQA uses to derive answers. The operators factor the end-to-end QA problem into smaller subproblems:

Parsing operators (Section 5.7.1) are responsible for interfacing between natural language questions and the KB query language described in Section 5.3.2. OQA uses a small number of high-precision templates to map questions to queries.

Paraphrase operators (Section 5.7.2) are responsible for rewording the input question into the domain of a parsing operator. In an offline process, OQA mines 5 million lexicalized paraphrase-templates from an unlabeled corpus of open-domain questions.

Query-rewrite operators (Section 5.7.3) are responsible for interfacing between the vocabulary used in the input question and the internal vocabulary used by the KBs. OQA implements its query-rewrite operators by mining a set of 75 million relation-entailment pairs from the knowledge bases described in Section 5.3.1.

The execution operator (Section 5.7.4) is responsible for fetching and combining evidence from the KB, given a query.

For each operator, OQA computes general, domain independent features that are used in the scoring function. These features are *unlexicalized* in the sense that they do not compute any values associated with content words in either the question or the KB.

In the following subsections, we describe each type of operator in detail and describe the operator-specific features used by the scoring function.

Question Pattern	**Query Pattern**	**Example Question**	**Example Query**
Who/What RV_{rel} NP_{arg}	`(?x, rel, arg)`	Who invented papyrus?	`(?x, invented, papyrus)`
Who/What Aux NP_{arg} RV_{rel}	`(arg, rel, ?x)`	What did Newton discover?	`(Newton, discover, ?x)`
Where/When Aux NP_{arg} RV_{rel}	`(arg, rel in, ?x)`	Where was Edison born?	`(Edison, born in, ?x)`
Where/When is NP_{arg}	`(arg, is in, ?x)`	Where is Detroit?	`(Detroit, is in, ?x)`
Who/What is NP_{arg}	`(arg, is-a, ?x)`	What is potassium?	`(potassium, is-a, ?x)`
What/Which NP_{rel2} Aux NP_{arg} RV_{rel1}	`(arg, rel1 rel2, ?x)`	What sport does Sosa play?	`(Sosa, play sport, ?x)`
What/Which NP_{rel} is NP_{arg}	`(arg, rel, ?x)`	What ethnicity is Dracula?	`(Dracula, ethnicity, ?x)`
What/Who is NP_{arg}'s NP_{rel}	`(arg, rel, ?x)`	What is Russia's capital?	`(Russia, capital, ?x)`
What/Which NP_{type} Aux NP_{arg} RV_{rel}	`(?x, is-a, type) (arg, rel, ?x)`	What fish do sharks eat?	`(?x, is-a, fish) (sharks, eat, ?x)`
What/Which NP_{type} RV_{rel} NP_{arg}	`(?x, is-a, type) (?x, rel, arg)`	What states make oil?	`(?x, is-a, states) (?x, make, oil)`

Table 5.2: High-precision parsing operators used to map questions to queries. Question templates are expressed using noun phrases (NP), auxiliary verbs (Aux), and ReVerb patterns (RV). Subscripts denote regex-style capture groups.

5.7.1 Parsing Operators

To map questions to queries, we use the set of 10 hand-written operators shown in Table 5.2. Each operator consists of a question pattern and a query pattern.

A question pattern is expressed as a regular expression over part-of-speech (POS) tags and function words. To detect noun phrases (NPs), we use a POS pattern that matches a sequence of nouns, determiners, and adjectives. We use the ReVerb pattern (Fader et al., 2011) to detect relation phrases. Each question pattern uses named capture-groups to select substrings from the input question.

A query pattern consists of a query (defined in Section 5.3.2) containing pointers to capture groups from the question pattern. When a question pattern matches a question, the captured strings are substituted into the query pattern to generate a complete query. For example, the question pattern in the first row of Table 5.2 matches "Who invented papyrus?" and captures the substrings {rel → "invented", arg → "papyrus"}. These are substituted into the query pattern `(?x, rel, arg)` to produce the output `(?x, invented, papyrus)`.

Features: Because some question patterns may be more reliable than others, we include an indicator feature for each question pattern. We also include indicator features for the POS sequence of the capture groups and the POS tags to the left and right of each capture group.

5.7.2 Paraphrasing Operators

The parsing operators in Table 5.2 have high precision but low recall. To increase recall, we use paraphrase operators to map questions onto the domain of the parsing operators. Each paraphrase operator is implemented as a pair of paraphrase templates like the examples in Table 5.3.

Each paraphrase template consists of a natural language string with a slot that captures some argument.[3] For example, the first source template in Table 5.3 matches the question

[3]While Paralex allows for both 1-argument and 2-argument templates, oqa only uses 1-argument templates. Extending oqa to use 2-argument templates is future work.

Source Template	Target Template
How does _ affect your body?	What body system does _ affect?
What is the latin name for _?	What is _'s scientific name?
Why do we use _?	What did _ replace?
What to use instead of _?	What is a substitute for _?
Was _ ever married?	Who has _ been married to?

Table 5.3: Example paraphrase operators that extracted from a corpus of unlabeled questions.

"How does nicotine affect your body?" This question can then be paraphrased by substituting the argument "nicotine" into the target template, yielding the new question "What body system does nicotine affect?"

We follow the work of PARALEX and automatically mine these source/target template pairs from the WikiAnswers[4] paraphrase corpus. We use an updated crawl of WikiAnswers that consists of 23 million question-clusters that users have grouped as synonymous. Each question cluster contains an average of 25 questions. In general, the clusters have low precision due to mistakes or users grouping related, but non-synonymous questions (*e.g.*, "How to say shut up in french?" is grouped with "Is it nice to say shut up?").

We extracted 200,000 templates that occurred in at least 10 question clusters. For each pair of templates (t, t'), we define the co-occurrence count $c(t, t')$ to be the number of clusters where t and t' both occur with the same argument. For example, if a cluster contains the questions "Why do we use computers?" and "What did computers replace?" we would increment the count c(Why do we use _?, What did _ replace?) by 1. For each template pair (t, t') such that $c(t, t') \geq 5$, we define paraphrase operators $t \rightarrow t'$ and $t' \rightarrow t$. This generates a set of 5 million paraphrase operators. During inference, all possible paraphrases of a question q are computed by considering all substrings of q (up to 5 tokens) as the argument.

[4] `http://wiki.answers.com`

Source Query	Target Query
(?x, children, ?y)	(?y, was born to, ?x)
(?x, birthdate, ?y)	(?x, date of birth, ?y)
(?x, is headquartered in, ?y)	(?x, is based in, ?y)
(?x, invented, ?y)	(?y, was invented by, ?x)
(?x, is the language of, ?y)	(?y, languages spoken, ?x)

Table 5.4: Example query-rewrite operators mined from the knowledge bases described in Section 5.3.1.

Features: The paraphrase operators are automatically extracted from a noisy corpus, and are not always reliable. We compute statistical and syntactic features to estimate the confidence of using the operator $t \rightarrow t'$ to paraphrase a question q to a new question q' using argument a. The statistical features include the pointwise mutual information (PMI) between t and t' in the WikiAnswers corpus and a language model score of q'. The syntactic features include the POS sequence of the matched argument a, and the POS tags to the left and right of a in q.

5.7.3 Query-Rewrite Operators

To handle the mismatch between natural language vocabulary and the KB vocabulary, we mine query rewrite rules. We focus on handling the mismatch between relation words in the question and relation words in the KB.[5] Table 5.4 lists example query rewrite rules. Each rule encodes a translation from one relation phrase to another, with a possible re-ordering of the arguments. For example, the first row in Table 5.4 is an operator that allows the relation phrase `children` to be rewritten as `was born to`$^{-1}$, where the superscript denotes inverted argument ordering.

We follow previous work on mining equivalent relations (Lin and Pantel, 2001; Yates and Etzioni, 2009) and count the number of shared argument-pairs between two relation phrases.

[5]Rewriting arguments is future work.

For example, the tuples `(hermann einstein, children, albert einstein)` and `(albert einstein, was born to, hermann einstein)` both appear in the KB, so `children` and `was born to`$^{-1}$ share the argument pair `(hermman einstein, albert einstein)`. We construct a query rewrite operator for each pair of relation phrases (r, r') that share at least 10 argument pairs. This results in a set of 74 million (r, r') pairs that we include as operators.

Features: As with the paraphrase templates (Section 5.7.2), we compute the PMI for each pair of relation phrases as a feature.

5.7.4 Execution Operator

The execution operator takes a query as input and returns a set of tuples, as shown in Figure 5.2. We store the KB `(arg1, relation, arg2)` triples in an inverted index[6] that allows for efficient keyword search over the three triple fields. We implemented a simple query optimizer that performs joins over triples by making multiple queries to the KB. Due to the size of the KB, we limit each keyword search over the triples to return the top 100 hits. The output of the execution operator is an answer state, containing a string answer and a joined tuple of evidence.

Features: We use features to estimate the reliability of the KB output. The features examine properties of the query, the returned tuple evidence, and the answer.

We measure the keyword similarity between two strings by lemmatizing them, removing stopwords, and computing the cosine similarity. We then include the keyword similarity between the query and the input question, the keyword similarity between the query and the returned evidence, and an indicator feature for whether the query involves a join.

The evidence features compute KB-specific properties. Extracted triples have confidence scores, which are included as features. We compute the join-key string similarity measured using the Levenshtein distance. We also include indicator features for the source of each triple (*e.g.*, whether the triple is from Open IE or Freebase).

The answer features compute conjunctions of properties of the input question and the

[6] `https://lucene.apache.org/solr/`

answer. We compute whether the question begins with some common prefixes (*e.g.*, Who, What, When, How many, *etc.*). For the answer, we compute word-shape features (*e.g.*, "Kansas" has the word shape "Aaaa" and "December 1941" has the word shape "Aaaa 1111"). This allows the system to learn that features like *question starts with When* $\wedge$ *answer has shape 1111* are indicative of a correct answer.

5.8 Experimental Setup

We are interested in answering three questions: (1) How does OQA compare to the state-of-the-art systems PARALEX, SEMPRE, and Wolfram|Alpha? (2) How do the different knowledge sources affect performance? (3) How do the different system components affect performance?

We investigate these questions by comparing performance on three question sets. Given a question q, each system returns an answer a with confidence $c \in \mathbb{R}$ or "no answer." We then measure the precision (correct answers/answers returned), recall (correct answers/questions), and F1 score (harmonic mean of precision and recall). We also compute precision-recall curves that show how precision is traded for recall as the minimum confidence to return an answer is varied. We describe the three question sets in Section 5.8.1 and the system settings in Section 5.8.2.

5.8.1 Question Sets

In our experiments, we use three question sets: WebQuestions, TREC, and WikiAnswers. Figure 5.5 shows statistics and example questions from each set.

WebQuestions was introduced by the authors of the SEMPRE system (Berant et al., 2013). The questions were generated from Google Autocomplete using a seed set of Freebase entities. Amazon Mechanical Turk users then provided answers in the form of Freebase concepts. Questions that could not be answered using Freebase were filtered out. Out of the three test sets, WebQuestions has the unique property that the questions are known *a priori* to be answerable using Freebase.

WebQuestions	who influenced wolfgang amadeus mozart?
3,778 train	who won the super bowl xliv 2010?
2,032 test	where was nicki minaj born?
	what is in liverpool england?
	who is the leader of france 2012?
TREC	What other countries do Kurds live in?
962 train	What year was Barry Manilow born?
517 test	What format was VHS's main competition?
	Who is the chairman of WWE?
	What is Muscular Dystrophy?
WikiAnswers	What is Matthew henson's mothers name?
1,334 train	Who is a retired gay nfl football player?
731 test	Do beluga whales eat penguins?
	Why were the conquistadors important?
	How does psychiatry benefit society?

Table 5.5: The three question sets used in our experiments.

TREC was introduced for the purpose of evaluating information retrieval QA systems (Voorhees and Tice, 2000). We re-purpose the TREC questions to test our KB-based Open QA systems. While the TREC questions were designed to be answerable using a small collection of test documents, they are not guaranteed to be answerable using any of the KBs described in Section 5.3.1.

WikiAnswers is a set of questions that were randomly sampled from a crawl of WikiAnswers. The WikiAnswers question set is disjoint from the corpus used to extract the paraphrasing operators described in Section 5.7.2. WikiAnswers questions are very challenging and ambiguous, and are not necessarily answerable by any KB.

WebQuestions and TREC both have gold-standard answer sets for each question, and WikiAnswers questions often have no answers available. However, due to the open-domain nature of our experiments, the gold-standard answer sets are incomplete. If a system's top answer was not already included in the provided gold-standard sets, we manually tagged the answers as correct or incorrect.

5.8.2 System Settings

OQA: We examine two different training conditions. In the first condition, we trained OQA on each training set independently. In the second condition, we trained OQA on the union of the WebQuestions, TREC, and WikiAnswers training sets.

For inference, we used a beam capacity of 1,000 and a search time limit of 20 seconds. For learning, we initialized 10 of the feature weights to be +1/-1 based on whether the features are indicative of good derivations (*e.g.*, PMI scores) or bad derivations (*e.g.*, verbs as paraphrase-template arguments). We set the number of perceptron iterations (between 1 and 5) using a fraction of held-out training data. For the first perceptron iteration, we interactively trained the system by providing the set D^* in Figure 5.4.

Paralex: We used PARALEX to parse questions to queries, and then execute them against the same KB as OQA. PARALEX provides a score for each query. For each answer that the query returns, we use the score from the query as a measure of confidence.

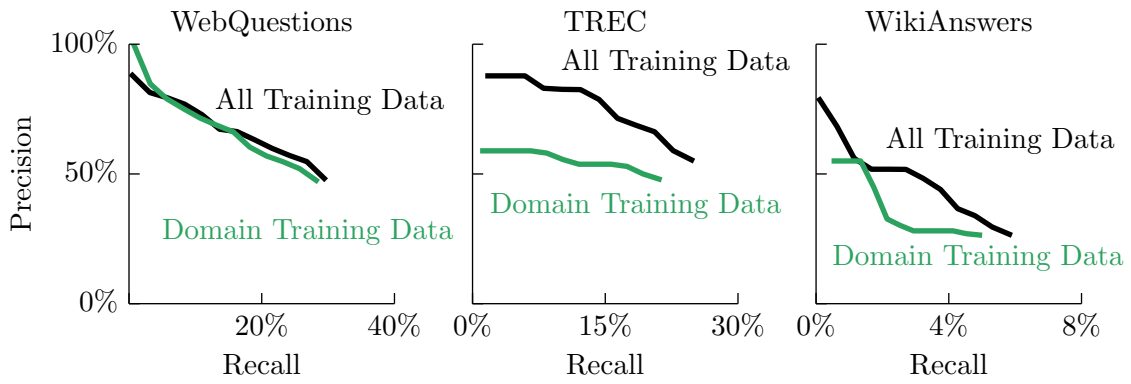


Figure 5.5: Training the scoring function on the union of all training data results in higher precision and recall on TREC and WikiAnswers.

Sempre: The authors of SEMPRE also make it available for download.[7] SEMPRE comes with a model trained on the WebQuestions question set. We attempted to train SEMPRE with questions from TREC and WikiAnswers, but found that the WebQuestions model had higher performance on held-out development questions, so we use the WebQuestions model in our experiments. The SEMPRE system is only able to access Freebase, and is thus used as a baseline to test the OQA's ability to access both curated and extracted KBs.

Wolfram|Alpha: We queried Wolfram|Alpha[8] via its public API. We took the top answer returned by the system.

5.9 Experimental Results

Figure 5.5 shows the precision-recall curves comparing OQA under the two different training conditions. Training the scoring function on the union of all training data results in higher precision and recall on TREC and WikiAnswers, but does not have a large effect on the WebQuestions performance. TREC and WikiAnswers contain many questions that cannot

[7] `https://github.com/percyliang/sempre`

[8] `http://www.wolframalpha.com`

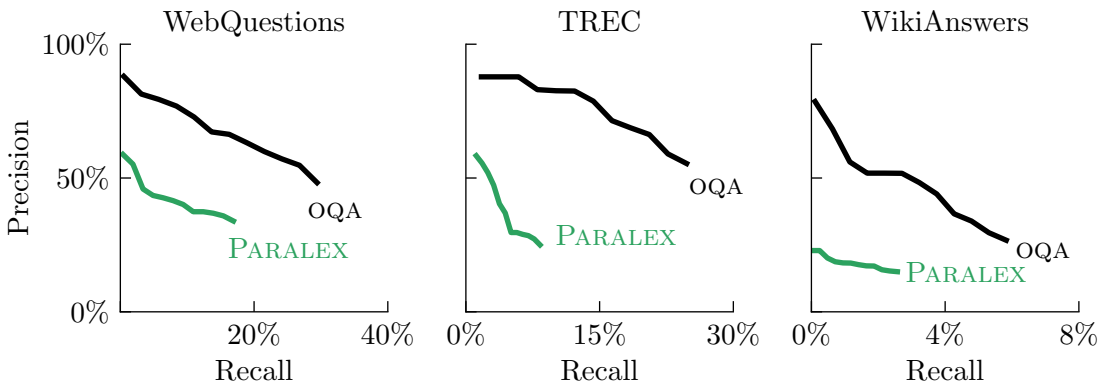


Figure 5.6: OQA has higher precision and recall than the Open QA system PARALEX.

be answered using the OQA knowledge base, so the effective size of the training sets is smaller than WebQuestions. The questions in WebQuestions are known to be answerable using Freebase, so OQA is able to learn from a greater fraction of the WebQuestions training set than on TREC and WikiAnswers.

Figure 5.6 shows the precision-recall curves of OQA and PARALEX on the test questions. OQA achieves both higher precision and recall than PARALEX across all three question sets. OQA's scoring function was able to avoid many of the errors made by PARALEX. For example, PARALEX made a systematic error confusing "Where" and "When" questions, *e.g.*, it was unable to tell that `1985` is an unlikely answer to a question that begins with "Where." In contrast, OQA was able to compute features of the full derivation (including the answer), which allowed it to learn not to make this type of error.

Figure 5.7 shows the precision-recall curves of OQA and SEMPRE on the test questions. In this case, SEMPRE is has higher precision and recall than OQA on WebQuestions. SEMPRE performs better on WebQuestions through its use of lexicalized features, *e.g.*, there is a single feature indicating that the string "see in" corresponds to the Freebase relation `tourist attraction`. These features allow SEMPRE to better fit the distribution of relations and entities in WebQuestions. In contrast, OQA uses only unlexicalized features like POS tags and corpus statistics like PMI, which limit OQA's ability to fit the WebQuestions training

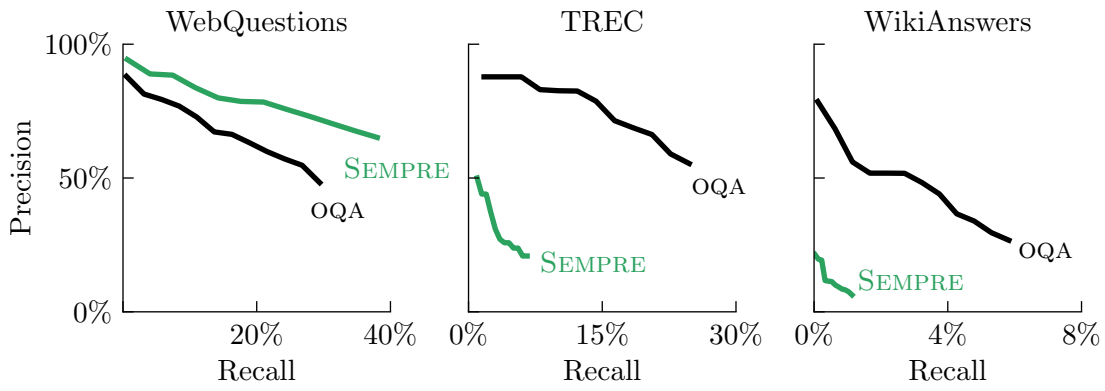


Figure 5.7: SEMPRE has higher precision and recall on WebQuestions, which are known to be answerable in Freebase. However, OQA outperforms SEMPRE on TREC and WikiAnswers, which were not developed for any particular KB.

data.

However, OQA performs better on TREC and WikiAnswers for two reasons. First, OQA is uses both extracted and curated knowledge sources, so it is more likely to have an answer in its KB. Second, SEMPRE requires lexical overlap in its training and testing set, which is not satisfied in the TREC and WikiAnswers questions.

Figure 5.8 shows the precision-recall curves of OQA compared to Wolfram|Alpha (WA). Because WA does not return a confidence value for its answers, we plot a single precision-recall point. WA has higher precision than OQA, but lower recall. WA uses "extensive human curation" to perform question interpretation at very high precision.[9] However, it fails to answer simple questions like "What do the Maori Eat?" that are not covered by its curated knowledge base.

Figure 5.9 shows the effects of removing different components from OQA. The weight-learning algorithm improves performance over the default weights defined in the experimental setup, with the exception of WikiAnswers. The learned weights have lower F1 on WikiAnswers because most questions in the WikiAnswers training set could not be parsed

[9] `http://www.wolfram.com/natural-language-understanding`

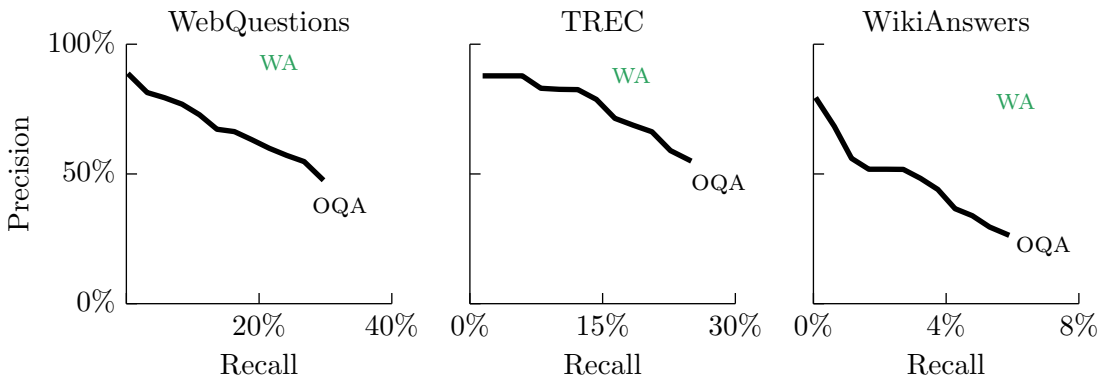


Figure 5.8: Wolfram|Alpha achieves higher precision but lower recall than OQA.

to any answer, which reduced the effective size of the training set.

The paraphrase operators improve performance on WebQuestions and WikiAnswers, but not on TREC. We found that many TREC questions were covered by the parser operators in Table 5.2, so the paraphrase operators did not add much. In contrast, the WebQuestions and WikiAnswers questions exhibit much more lexical and syntactic variation, so the paraphrase operators helped.

The query-rewrite operators led to only a slight improvement on the TREC question set, and had at best no effect on WebQuestions and WikiAnswers. We examined the output and found some high-confidence examples where the query-rewrite operators helped, *e.g.*, the question "When did the Big Dig begin?" was answered by rewriting `(big dig, begin in, ?x)` to `(big dig, broke ground in, ?x)`. However, most derivations that used a query-rewrite operator were assigned low confidence, and had limited effect on recall.

Figure 5.10 shows the effects of removing a knowledge source from OQA on system performance. Removing Open IE from the KB lowers the F1 score across all test sets. Freebase helps the most on the WebQuestions set (which was designed specifically for Freebase), but is less useful for TREC and WikiAnswers. Probase is most useful for WikiAnswers, which contains many "What is…" questions that can be answered using Probase's is-a relations. NELL is largely a subset of the other KBs, so it had no effect on OQA's performance.

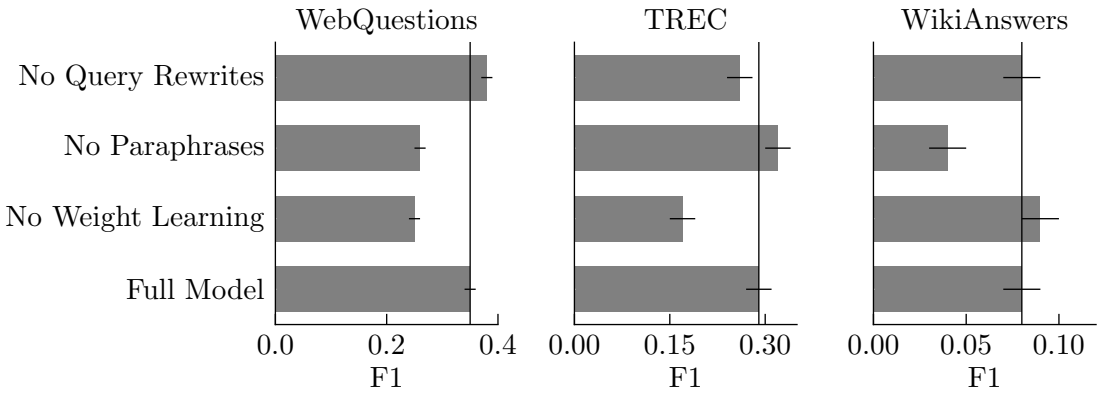


Figure 5.9: The relative contributions of each system component depend on the distribution of test questions. (Error bars represent one standard deviation from the mean, computed over 10,000 bootstrap samples of test data.)

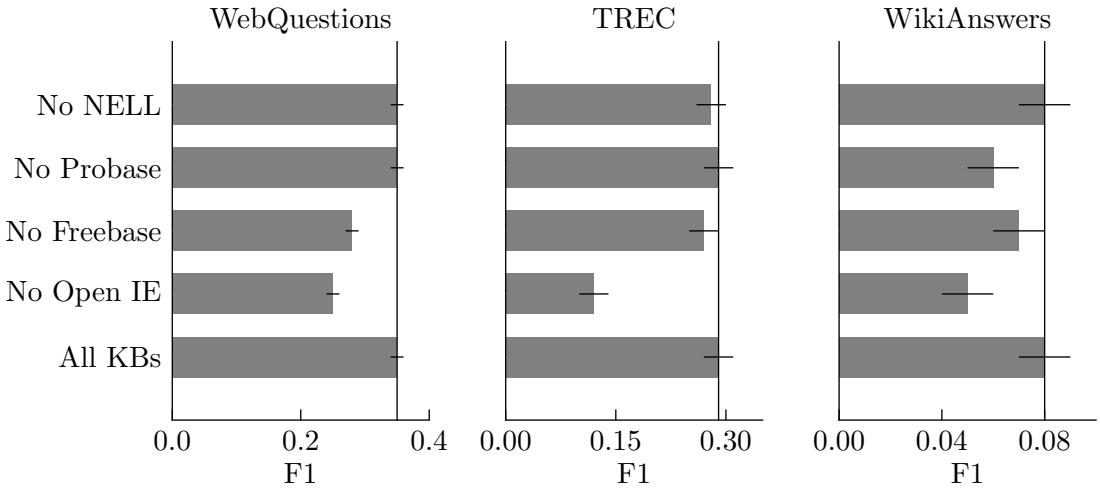


Figure 5.10: OQA performs best using multiple knowledge sources, in particular Open IE, Freebase, and Probase.

5.10 Discussion

The experimental results in the previous section exemplify the power-generality tradeoff discussed in the introduction: OQA uses a small number of general, unlexicalized features, which provided better generalization. However, this limits OQA's ability to take full advantage of the training data. For example, OQA was unable to answer questions like "What time zone is in South Africa have?" despite seeing several nearly-identical questions in the WebQuestions training data. A challenge for the future is to engineer OQA to take advantage of lexical cues when they are available. Extending OQA to induce higher-precision operators during discriminative training may be one way.

One problem that has gone unaddressed by all of the discussed QA systems is modeling whether a given question is answerable with the given KB and operators. For example, OQA currently has no way to answer truth-false questions like "Are dogs mammals?" Yet OQA systematically chains low-confidence operators together to derive incorrect answers for them, which hurts precision. A measure of answerability would be useful in scenarios where high-precision is required.

Table 5.6 shows examples from the test data where OQA derives correct answers. The first example shows a query rewrite operator that modifies the relation `marry` to `has wife`. The second example shows a paraphrase operator that maps "What are _ made of?" to "What material are _ made of?" In this case, the paraphrase operator introduces a type constraint that does not appear in the input question, which is beneficial for selecting the correct answer. The third example highlights the benefit of extracted knowledge, which contains obscure assertions like "`(changing light bulb, is-a, small building maintenance job)`.

Table 5.7 shows examples where OQA derives incorrect answers. The first example shows that the paraphrase operators can be too general, in this case overgeneralizing "animal" to "symbol." This combines with `California Water Service` incorrectly matching `California`, resulting in the incorrect answer `CWT`. The second example shows two types of errors. First, the parser incorrectly treats "actor first" as an argument to a type constraint. Then, the execution operator (which ignores word order in keyword matching) matches

Operator	State
Input	Who did Michael J Fox marry?
Parse	`?x: (Michael J Fox, marry, ?x)`
Rewrite	`?x: (Michael J Fox, has wife, ?x)`
Execute	`Tracy Pollan:` `(Michael J. Fox, has wife, Tracy Pollan)`
Input	What are brake pads made of?
Paraphrase	What material are brake pads made of?
Parse	`?x: (?x, is-a, material) (brake pads, made of, ?x)`

Execute	`copper:`
	`(copper, is-a, material)`
	`(The brake pads, were made of, copper)`

Input	What are some examples of building maintenance jobs?
Parse	`?x: (?x, example of, building maintenance jobs)`
Rewrite	`?x: (?x, is-a, building maintenance jobs)`

Execute	`changing light bulb:`
	`(changing light bulb, is-a,`
	`small building maintenance job)`

Table 5.6: Examples from the test data where OQA derives a correct answer.

Operator	State
Input	What animal represents California?
Paraphrase	What are California's symbols?
Parse	`?x: (california, symbols, ?x)`
Execute	**CWT:**
	`(California Water Service, Trading symbol, CWT)`
Input	What actor first portrayed James Bond?
Parse	`?x: (?x, is-a, actor first)`
	`(?x, portrayed, James Bond)`
Execute	`Daniel Craig:`
	`(Daniel Craig, is-a, first class actor)`
	`(Danny Craig, portrays, James Bond)`
Input	Who did George Washington admire?
Parse	`?x: (George Washington, admire, ?x)`
Execute	`presidents and generals:`
	`(George Washington, was admired by,`
	`presidents and generals)`

Table 5.7: Example derivations from the test data where OQA derives an incorrect answer.

`actor first` with `first class actor`, resulting in the incorrect answer `Daniel Craig`. The third example shows that better features are needed to prevent errors like matching an active voice ("admire") with the passive voice ("was admired by").

5.11 Conclusion

We introduced OQA, a novel Open QA system that is the first to leverage both curated and extracted knowledge. We described inference and learning algorithms that OQA uses to derive high-confidence answers. Our experiments demonstrate that OQA generalizes well to unseen questions and makes improvements over a state-of-the-art Open QA baseline.

Chapter 6

CONCLUSION

This dissertation focused on solving the problem of Open QA, which aims to satisfy complex information needs that are currently underserved by today's search engines. I described two major challenges for Open QA: knowledge acquisition and question interpretation. Chapter 2 discussed how the dominant approaches to question answering, answer retrieval and semantic parsing, are insufficient for Open QA. Answer retrieval systems are unable to answer complex questions that combine multiple pieces of evidence to derive an answer, while semantic parsing systems only work with small, single-domain knowledge bases.

My approach to Open QA is driven by my thesis statement that large-scale, open-domain information extraction can provide the necessary knowledge for Open QA, while community QA sites like WikiAnswers can be used to inform question interpretation. I presented three technical contributions, which attacked the problems of knowledge acquisition and question interpretation. The first contribution is the ReVerb Open IE system, which can be used to automatically extract a large, open-domain knowledge base from web text. ReVerb achieves higher precision and recall than previous approaches, while using a simpler model. The second contribution is the Paralex QA system, which uses the ReVerb knowledge base to answer questions. Paralex was the first Open QA system to use an extracted knowledge base, and leveraged the data from WikiAnswers using novel learning algorithms. The third and final contribution was the oqa system, which leveraged both extracted and curated knowledge bases for Open QA. oqa uses an accurate and robust scoring model, leading to higher performance than Paralex across multiple question sets.

This dissertation has presented first steps towards Open QA, but there is much work to be done. The rest of this chapter presents some open problems and ideas for future work on Open QA.

6.1 Machine Learning for Open QA

One future direction would be to enhance the learning algorithms used in the OQA system to achieve higher precision and recall. Some specific avenues that could be investigated include:

- **Modeling Answerability:** The OQA system has no way to judge whether a question is answerable given its knowledge. For example, OQA tries to answer the question "Should I remove my own wisdom teeth?" despite the fact that it is unlikely to be answerable under any knowledge base. This behavior can lower precision, since the system can often find a chain of operators leading to *some* answer. Explicitly modeling answerability could help increase precision by returning "no answer" for questions that are unlikely to be answerable.

- **Weak Supervision:** During learning, OQA updates its weights when its predicted answer is not in the ground-truth answer set, using exact string equality to evaluate set membership. This strict equality prevents OQA from using the free-text answers from WikiAnswers as a source of feedback. If OQA's learning algorithm could be modified to use indirect sources of feedback, then this would allow the system to learn from the millions of example question-answer pairs from WikiAnswers and elsewhere. One possible direction would be to adapt the loss-driven learning that has been explored in single-domain semantic parsing (Artzi and Zettlemoyer, 2013).

6.2 Compositional Analysis of Questions

One major limitation of the OQA system is its reliance on templates for paraphrasing and question parsing. For simple questions, the template approach seems to have adequate coverage. However, as questions get longer and more complex, the chance of having the right combination of paraphrase and parsing templates decreases. For example, none of OQA's templates match the "What is illegal in the US but legal in Mexico?" question from Chapter 1. A complete solution to this problem requires performing a compositional

semantic analysis of input questions, as is done in semantic parsing. However, combining this with the paraphrasing approach of PARALEX and OQA is not straightforward.

Another challenge for handling more complex questions is that they are underrepresented in naturally occurring question sources like WikiAnswers, where the most frequent questions are the single-relation questions that can be answered using templates. Thus, a major research contribution would be the creation of an evaluation set of questions that are both open-domain and require a compositional analysis in order to derive a correct answer.

6.3 Question-Guided Information Extraction

In Chapter 2, I discussed two approaches to constructing a knowledge base: manual curation or automatic extraction. Both of these approaches suffer from the drawback that they are uninformed by the actual information needs of people. Curated knowledge bases reflect the biases of its curators; extracted knowledge bases reflect the biases of the corpus they were extracted from (Gordon and Van Durme, 2013). Both of these approaches may be out of touch with what questions people actually have.

A complete Open QA system should be able to identify when its knowledge base is incomplete and automatically attempt to extend it. While previous approaches to automatic knowledge base completion were limited to extracting new instances of a known relation, the input distribution to a QA system could be used to target new relationships that do not exist within the knowledge base. For example, if enough people ask questions like "Is (athlete) left-handed or right-handed?" the system could infer that it should extend its knowledge base to have a relationship whose domain is athletes and whose range is either "left-handed" or "right-handed." A similar approach has been used for extracting attribute-value pairs using a search engine query log (Pasca and Van Durme, 2007), there has been no end-to-end QA system that modifies its knowledge in this way. The input questions to an Open QA system could serve as a strong and useful bias that is currently missing from Open Information Extraction techniques.

BIBLIOGRAPHY

Alan Akbik and Alexander Löser. Kraken: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 52–56. Association for Computational Linguistics, 2012.

David J. Allerton. *Stretched Verb Constructions in English.* Routledge Studies in Germanic Linguistics. Routledge (Taylor and Francis), New York, 2002.

Gabor Angeli and Christopher D Manning. Philosophers are mortal: Inferring the truth of unseen facts. *CoNLL-2013*, 133, 2013.

Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62, 2013.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, 1998.

Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni, et al. Rel-grams: a probabilistic model of relations in text. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 101–105. Association for Computational Linguistics, 2012.

M. Banko and O. Etzioni. The tradeoffs between traditional and open relation extraction. In *Proceedings of ACL*, 2008.

Michele Banko, Eric Brill, Susan Dumais, and Jimmy Lin. AskMSR: Question answering using the worldwide web. In *2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, 2002.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open Information Extraction from the Web. In *IJCAI*, 2007.

Colin Bannard and Chris Callison-Burch. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 2005.

Regina Barzilay and Lillian Lee. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics*, 2003.

Regina Barzilay and Kathleen R. McKeown. Extracting Paraphrases from a Parallel Corpus. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, 2001.

J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*, 2013.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. Global learning of typed entailment rules. In *Proceedings of ACL*, Portland, OR, 2011.

Jonathan Berant, Ido Dagan, Meni Adler, and Jacob Goldberger. Efficient tree-based approximation for entailment graph learning. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 117–125. Association for Computational Linguistics, 2012.

Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee. Linked data on the web. In *Proceedings of the 17th international conference on World Wide Web*, pages 1265–1266. ACM, 2008.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

Satchuthananthavale Rasiah Kuhan Branavan. *Grounding linguistic analysis in control applications*. PhD thesis, Massachusetts Institute of Technology, 2012.

Andrei Broder. A taxonomy of web search. In *ACM Sigir forum*. ACM, 2002.

John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George Miller, et al. Issues, tasks and program structures to roadmap research in question & answering (q&a). In *Document Understanding Conferences Roadmapping Documents*, pages 1–35, 2001.

Michael John Cafarella. *Extracting and managing structured web data*. PhD thesis, University of Washington, 2009.

Qingqing Cai and Alexander Yates. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *ACL*, 2013.

Chris Callison-Burch. Syntactic Constraints on Paraphrases Extracted from Parallel Corpora. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008.

Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM, 1998.

A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr., and T.M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, 1977.

Eugene Charniak. Toward a model of children's story comprehension. Technical report, DTIC Document, 1972.

Yang Chen and Daisy Zhe Wang. Knowledge expansion over probabilistic knowledge bases. In *SIGMOD*, 2014.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 52–60, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://portal.acm.org/citation.cfm?id=1866775.1866782`.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. Towards coherent multi-document summarization. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2013)*, 2013.

Jennifer Chu-Carroll, Krzysztof Czuba, John M Prager, Abraham Ittycheriah, and Sasha Blair-Goldensohn. Ibm's piquant ii in trec 2004. In *TREC*, 2004.

Jennifer Chu-Carroll, James Fan, BK Boguraev, David Carmel, Dafna Sheinwald, and Chris Welty. Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development*, 56(3.4):6–1, 2012.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. Driving Semantic Parsing from the World's Response. In *CoNLL*, 2010.

Michael Collins. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002.

Mark Craven, Andrew McCallum, Dan PiPasquo, Tom Mitchell, and Dayne Freitag. Learning to extract symbolic knowledge from the world wide web. Technical report, DTIC Document, 1998.

Jon Curtis, Gavin Matthews, and David Baxter. On the effective use of cyc in a question answering system. In *Proc Workshop on Knowledge and Reasoning for Answering Questions*, pages 61–70, 2005.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190. Springer, 2006.

Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the workshop on Human Language Technology*, pages 43–48. Association for Computational Linguistics, 1994.

Dipanjan Das and André FT Martins. A survey on automatic text summarization. *Literature Survey for the Language and Statistics II course at CMU*, 4:192–195, 2007.

AnHai Doan, Alon Y. Halevy, and Zachary G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

Doug Downey, Oren Etzioni, and Stephen Soderland. A probabilistic model of redundancy in information extraction. In *IJCAI*, pages 1034–1041, 2005.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of the 13th international conference on World Wide Web*, pages 100–110. ACM, 2004.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. Open information extraction: The second generation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pages 3–10. AAAI Press, 2011.

Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Paraphrase-Driven Learning for Open Question Answering. In *ACL*, 2013.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Submitted to KDD*, 2014.

James Fan, Aditya Kalyanpur, DC Gondek, and David A Ferrucci. Automatic knowledge extraction from documents. *IBM Journal of Research and Development*, 56(3.4):5–1, 2012.

Christiane Fellbaum. *WordNet*. Wiley Online Library, 1999.

D. A. Ferrucci. Introduction to "this is watson". *IBM J. Res. Dev.*, 56(3):235–249, May 2012. ISSN 0018-8646. doi: 10.1147/JRD.2012.2184356. URL `http://dx.doi.org/10.1147/JRD.2012.2184356`.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296, 1999.

Noah S Friedland, Paul G Allen, Gavin Matthews, Michael Witbrock, David Baxter, Jon Curtis, Blake Shepard, Pierluigi Miraglia, Jurgen Angele, Steffen Staab, et al. Project halo: Towards a digital aristotle. *AI magazine*, 25(4):29, 2004.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. Dependency-based open information extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 10–18. Association for Computational Linguistics, 2012.

Qin Gao and Stephan Vogel. Parallel Implementations of Word Alignment Tool. In *Proc. of the ACL 2008 Software Engineering, Testing, and Quality Assurance Workshop*, 2008.

Jonathan Gordon and Benjamin Van Durme. Reporting bias and knowledge extraction. In *AKBC*, 2013.

Arthur C Graesser and Natalie K Person. Question asking during tutoring. *American educational research journal*, 31(1):104–137, 1994.

Gregory Grefenstette and Simone Teufel. Corpus-based method for automatic identification of support verbs for nominalizations. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pages 98–103, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. doi: http://dx.doi.org/10.3115/976973.976988.

Barbara J. Grosz, Douglas E. Appelt, Paul A. Martin, and Fernando C. N. Pereira. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. *Artificial Intelligence*, 32(2):173–243, 1987.

Vishal Gupta and Gurpreet Singh Lehal. A survey of text summarization extractive techniques. *Journal of Emerging Technologies in Web Intelligence*, 2(3), 2010.

Lynette Hirschman and Robert Gaizauskas. Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275–300, 2001.

Lynette Hirschman, Marc Light, Eric Breck, and John D Burger. Deep read: A reading comprehension system. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 325–332. Association for Computational Linguistics, 1999.

Ben Hixon and Rebecca J Passonneau. Open dialogue management for relational databases. In *Proceedings of NAACL-HLT*, pages 1082–1091, 2013.

Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 286–295, 2010.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 2011.

Peter Jansen, Mihai Surdeanu, and Peter Clark. Discourse complements lexical semantics for non-factoid answer reranking. In *ACL*, 2014.

Aditya Kalyanpur, J William Murdock, James Fan, and Christopher Welty. Leveraging community-built knowledge for type coercion in question answering. In *The Semantic Web–ISWC 2011*, pages 144–156. Springer, 2011.

Gjergji Kasneci, Maya Ramanath, Fabian Suchanek, and Gerhard Weikum. The yago-naga approach to knowledge discovery. *ACM SIGMOD Record*, 37(4):41–47, 2009.

Holmer Hemsen Kathrin Eichler and Gnter Neumann. Unsupervised relation extraction from web documents. In *LREC*, 2008. ISBN 2-9517408-4-0. http://www.lrec-conf.org/proceedings/lrec2008/.

Boris Katz. Annotating the World Wide Web using Natural Language. In *RIAO*, pages 136–159, 1997.

Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy J. Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. Omnibase: Uniform access to heterogeneous data for question answering. In *Proceedings of the 6th International Conference on Applications of Natural Language to Information Systems-Revised Papers*, NLDB '02, pages 230–234, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-00307-X. URL `http://dl.acm.org/citation.cfm?id=645757.666145`.

Boris Katz, Sue Felshin, Jimmy J Lin, and Gregory Marton. Viewing the web as a virtual database for question answering. In *New Directions in Question Answering*, pages 215–226, 2004.

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.

Seokhwan Kim and Gary Geunbae Lee. A graph-based cross-lingual projection approach for weakly supervised relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 48–53. Association for Computational Linguistics, 2012.

Philipp Koehn. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *AMTA*, Lecture Notes in Computer Science, pages 115–124. Springer, 2004.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics*, 2003.

Julian Kupiec. Murax: A robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 181–190. ACM, 1993.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233. Association for Computational Linguistics, 2010.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*, 2013.

Cody Kwok, Oren Etzioni, and Daniel S. Weld. Scaling question answering to the web. *ACM Trans. Inf. Syst.*, 19(3):242–262, 2001.

Wendy G. Lehnert. A conceptual theory of question answering. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'77, pages 158–164, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc. URL `http://dl.acm.org/citation.cfm?id=1624435.1624467`.

Douglas B Lenat, Ramanathan V. Guha, Karen Pittman, Dexter Pratt, and Mary Shepherd. Cyc: toward programs with common sense. *Communications of the ACM*, 33(8):30–49, 1990.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Benjamin Taskar. An end-to-end discriminative approach to machine translation. In *ACL*, 2006.

Percy Liang, Michael Jordan, and Dan Klein. Learning Dependency-Based Compositional Semantics. In *ACL*, 2011.

Dekang Lin and Patrick Pantel. DIRT – Discovery of inference rules from text. In *KDD*, 2001.

Thomas Lin, Mausam, and Oren Etzioni. Identifying Functional Relations in Web Text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1266–1276, Cambridge, MA, October 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D10-1123`.

Thomas Lin, Mausam, and Oren Etzioni. Entity linking at web scale. In *AKBC-WEKEX*, 2012.

Xiao Ling and Daniel S Weld. Fine-grained entity recognition. In *AAAI*, 2012.

Vanessa Lopez, Victoria S. Uren, Enrico Motta, and Michele Pasin. Aqualog: An ontology-driven question answering system for organizational semantic intranets. *J. Web Sem.*, 5(2):72–105, 2007.

Paul Kingsbury Martha and Martha Palmer. From treebank to propbank. In *In Proceedings of LREC-2002*, 2002.

Yuval Marton, Chris Callison-Burch, and Philip Resnik. Improved Statistical Machine Translation Using Monolingually-Derived Paraphrases. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 2009.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to parse natural language commands to a robot control system. In *Proc. of the 13th International Symposium on Experimental Robotics (ISER)*, June 2012.

Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010.

Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. Effectiveness and efficiency of open relation extraction. In *EMNLP*, pages 447–457. ACL, 2013.

Scott Miller, Robert Bobrow, Robert Ingria, and Richard Schwartz. Hidden understanding models of natural language. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 25–32. Association for Computational Linguistics, 1994.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. A fully statistical approach to natural language interfaces. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 55–61. Association for Computational Linguistics, 1996.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant Supervision for Relation Extraction Without Labeled Data. In *ACL*, 2009.

Abdullah M Moussa and Rehab F Abdel-Kader. Qasyo: A question answering system for yago ontology. *International Journal of Database Theory & Application*, 4(2), 2011.

Vivi Nastase, Michael Strube, Benjamin Brschinger, Ccilia Zirn, and Anas Elghafari. Wikinet: A very large scale multi-lingual concept network. In *In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010*, pages 1015–1022, 2010.

Vincent Ng. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1396–1411, 2010.

Rodney D Nielsen, Jason Buckingham, Gary Knoll, Ben Marsh, and Leysia Palen. A taxonomy of questions for question generation. In *Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge*, 2008.

Franz Josef Och and Hermann Ney. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, 2000.

Marius Pasca and Benjamin Van Durme. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI*, volume 7, pages 2832–2837, 2007.

Roberto Pieraccini, Evelyne Tzoukermann, Zakhar Gorelov, J Gauvain, Esther Levin, Chin-Hui Lee, and Jay G Wilpon. A speech understanding system based on statistical repre-

sentation of semantics. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 193–196. IEEE, 1992.

A. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates. PRECISE on ATIS: Semantic tractability and experimental results. In *Procs. of the 19th National Conference on Artificial Intelligence (AAAI-04)*, pages 1026–1027, 2004.

John Prager. Open-domain question answering. *Found. Trends Inf. Retr.*, 1(2):91–231, January 2006. ISSN 1554-0669. doi: 10.1561/1500000001. URL `http://dx.doi.org/10.1561/1500000001`.

Vaughan Pratt. Cyc report. `http://boole.stanford.edu/cyc.html`, 1994.

V. Punyakanok, D. Roth, and W. Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2), 2008.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. Natural language inference via dependency tree mapping: An application to question answering. *Computational Linguistics*, 2004.

Chris Quirk, Chris Brockett, and William Dolan. Monolingual Machine Translation for Paraphrase Generation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.

Bertram Raphael. A computer program which understands. In *Proceedings of the October 27-29, 1964, fall joint computer conference, part I*, pages 577–589. ACM, 1964.

Deepak Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 41–47. Association for Computational Linguistics, 2002.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling Relations and Their Mentions without Labeled Text. In *Proceedings of the 2010 European conference on Machine learning and Knowledge Discovery in Databases*, 2010.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. In *HLT-NAACL*, 2013.

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.

Alan Ritter, Mausam, and Oren Etzioni. A Latent Dirichlet Allocation method for selectional preferences. In *ACL*, 2010.

Nicholas Roy, Joelle Pineau, and Sebastian Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 93–100. Association for Computational Linguistics, 2000.

Roger C Schank and Robert P Abelson. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press, 2013.

Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics, 2012.

Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld, and Jesse Davis. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1088–1098, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://portal.acm.org/citation.cfm?id=1870658.1870764`.

Lenhart Schubert. Can we derive general world knowledge from texts? In *Proceedings of the second international conference on Human Language Technology Research*, pages 94–97. Morgan Kaufmann Publishers Inc., 2002.

Satoshi Sekine. On-demand information extraction. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 731–738, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

Yusuke Shinyama and Satoshi Sekine. Preemptive Information Extraction using Unrestricted Relation Discovery. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 304–311, New York City, USA, June 2006. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N/N06/N06-1039`.

R. F. Simmons. Answering english questions by computer: A survey. *Commun. ACM*, 8 (1):53–70, January 1965. ISSN 0001-0782. doi: 10.1145/363707.363732. URL `http://doi.acm.org/10.1145/363707.363732`.

Robert F Simmons. Natural language question-answering systems: 1969. *Communications of the ACM*, 13(1):15–30, 1970.

S. Soderland. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1-3):233–272, 1999. URL `citeseer.ist.psu.edu/soderland99learning.html`.

Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu, Mausam, and Oren Etzioni. Adapting open information extraction to domain-specific relations. *AI Magazine*, 31(3):93–102, 2010.

Lucia Specia and Enrico Motta. M.: A hybrid approach for extracting semantic relations from texts. In *In. Proceedings of the 2 nd Workshop on Ontology Learning and Population*, pages 57–64, 2006.

Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.

M. Stevenson. An unsupervised WordNet-based algorithm for relation extraction. In *Proceedings of the "Beyond Named Entity" workshop at the Fourth International Conference on Language Resources and Evalutaion (LREC'04)*, 2004.

Suzanne Stevenson, Afsaneh Fazly, and Ryan North. Statistical measures of the semiproductivity of light verb constructions. In *In 2nd ACL Workshop on Multiword Expressions: Integrating Processing*, pages 1–8, 2004.

David G Stork. The open mind initiative. *IEEE Expert Systems and Their Applications*, 1999.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

Fabian M Suchanek, Mauro Sozio, and Gerhard Weikum. Sofie: a self-organizing framework for information extraction. In *Proceedings of the 18th international conference on World wide web*, pages 631–640. ACM, 2009.

Xu Sun, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. Latent variable perceptron algorithm for structured classification. In *IJCAI*, 2009.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191, 2008.

Yuen-Hsien Tseng, Lung-Hao Lee, Shu-Yen Lin, Bo-Shun Liao, Mei-Jun Liu, Hsin-Hsi Chen, Oren Etzioni, and Anthony Fader. Chinese open relation extraction for knowledge acquisition. *EACL 2014*, page 12, 2014.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-Based Question Answering over RDF Data. In *WWW*, 2012.

Rajan Vaish, Keith Wyngarden, Jingshu Chen, Brandon Cheung, and Michael S Bernstein. Twitch crowdsourcing: Crowd contributions in short bursts of time. In *CHI*, 2014.

Benjamin Van Durme and Lenhart Schubert. Open knowledge extraction through compositional language processing. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 239–254. Association for Computational Linguistics, 2008.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 75–80. Association for Computational Linguistics, 2007.

Ellen M. Voorhees and Dawn M. Tice. Building a question answering test collection. In *SIGIR*, 2000.

Sebastian Walter, Christina Unger, Philipp Cimiano, and Daniel Br. Evaluation of a Layered Approach to Question Answering over Linked Data. In *ISWC*, 2012.

Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language, 1971.

Yuk Wah Wong and Raymond J. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL*, 2007.

W. A. Woods. Progress in natural language understanding: An application to lunar geology. In *Proceedings of the June 4-8, 1973, National Computer Conference and Exposition*, AFIPS '73, pages 441–450, New York, NY, USA, 1973. ACM. doi: 10.1145/1499586.1499695. URL `http://doi.acm.org/10.1145/1499586.1499695`.

Fei Wu and Daniel S. Weld. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10)*, 2010.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD*, 2012.

Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. Open information extraction with tree kernels. In *Proceedings of NAACL-HLT*, pages 868–877, 2013.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. Natural Language Questions for the Web of Data. In *EMNLP*, 2012.

Limin Yao, Sebastian Riedel, and Andrew McCallum. Unsupervised Relation Discovery with Sense Disambiguation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 2012.

Alexander Yates and Oren Etzioni. Unsupervised methods for determining object and relation synonyms on the web. *JAIR*, 34:255–296, March 2009.

Naomi Zeichner. *Evaluating and Improving Inference-rules via Crowdsourcing.* PhD thesis, Bar-Ilan University, 2012.

John M. Zelle and Raymond J. Mooney. Learning to Parse Database Queries Using Inductive Logic Programming. In *AAAI*, 1996.

Luke S. Zettlemoyer and Michael Collins. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *UAI*, 2005.

Luke S Zettlemoyer and Michael Collins. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language*

Processing of the AFNLP: Volume 2-Volume 2, pages 976–984. Association for Computational Linguistics, 2009.

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. StatSnowball: a statistical approach to extracting entity relationships. In *WWW '09: Proceedings of the 18th international conference on World Wide Web*, pages 101–110, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: http://doi.acm.org/10.1145/1526709.1526724.