Interactive Learning of Relation Extractors with Weak Supervision

Raphael Hoffmann

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

Doctor of Philosophy

University of Washington

2012

Daniel S. Weld, Chair

Luke Zettlemoyer, Chair

Oren Etzioni

Program Authorized to Offer Degree:
UW Computer Science & Engineering

University of Washington

Abstract

Interactive Learning of Relation Extractors with Weak Supervision

Raphael Hoffmann

Co-Chairs of the Supervisory Committee:
Professor Daniel S. Weld
Computer Science & Engineering

Assistant Professor Luke Zettlemoyer
Computer Science & Engineering

The ability to automatically convert natural language text into a knowledge base may open the door to great new opportunities, including question-answering on the Web, detection of trends and sentiments in social media, and perhaps even intelligent agents which understand our language. Today, however, there does not exist a system that can reliably convert text into a knowledge base, and the task turns out to be far more difficult than it appears. A key challenge is *relation extraction* – detecting semantic relationships between entities mentioned in text. Most successful approaches use supervised machine learning, but creating the required labeled training examples has proven too expensive for constructing Web-scale knowledge bases.

This dissertation shows that we can greatly reduce the amount of human effort necessary to create relation extractors by leveraging a richer set of user interactions, some of which use more accurate models of *weak supervision* from a database. Specifically, this dissertation presents (1) a weakly supervised technique based on multi-instance learning which allows relations to overlap, (2) a weakly supervised technique that allows learning from only a few instances per relation by dynamically inducing relation-specific lexicons, (3) an approach for developing extraction rules interactively, and (4) a technique which synergistically pairs weakly supervised relation extraction with extraction validation by an online community. Our proposed techniques make it possible to create a high-quality relation extractor in under one hour, moving us closer towards automatically constructing Web-scale knowledge-bases.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I was fortunate to having had the chance to work with several outstanding collaborators, each of whom have taught me an enormous amount of skills, and each of whom gave me the confidence and passion to create new things that will perhaps one day make our world a little better.

First, I would like to thank Dan Weld. Dan has been incredibly supportive and enabled me to do research on a wide range of things. Today, it still baffles me how he can be so successful at so many things — whether it is AI, HCI, or NLP research, creating companies, or wandering through Utah's wilderness — and how he discovers new research directions which can make a big impact on our society. Furthermore, special thanks go to Luke Zettlemoyer, who has gotten me enthusiastic about natural language processing. Luke has an exciting vision for how semantic parsing, user interactions in natural language, and computer vision can all fit together. From Luke, I learned much about machine learning and language processing. Special thanks also go to James Fogarty, who not only triggered my interest in human-computer interaction, but also helped me get on track with research and publishing. James has the ability to make a paper so perfect that it leaves all reviewers in awe (yes, I am talking about our UIST07 paper).

Although I did not directly collaborate with Oren Etzioni and Pedro Domingos, their seminars were always a great source of new ideas, many of which shaped my work. What is fascinating about Oren is how he not only leads research which *might* impact millions of people, but occasionally develops it until it *does* impact millions of people. And every once in a while, Pedro impressed with some groundbreaking machine learning magic which then fully occupied my mind for several days.

Parts of this dissertation were done in collaboration with Saleema Amershi, Kayur Patel, Xiao Ling, Fei Wu, and Congle Zhang, whom I thank. We had a fun time together, and I learned many things from them.

DEDICATION

to my wife, Jing, for her support and encouragement

Chapter 1

INTRODUCTION

We are intrigued by the idea of automatically converting natural language text into a knowledge-base. When applied to the vast amounts of text on the Web, the possibilities seem endless: We could build a question-answering system that combines facts stated on different pages to give us answers when today's search engines fail. We could perform analytical experiments that help us detect sentiments and trends, giving us new insights into society and human behavior. We could also aggregate and visualize experimental results from thousands of research publications to accelerate progress in drug discovery. We might even get an important step closer to creating artificial intelligence, since the technology might allow us to build intelligent agents that we could talk to in natural language.

Today, however, there is no system that can reliably convert text into a knowledge-base, and the task is far more difficult than it seems. For example, it is not even clear how to store knowledge. One idea is to explicitly specify a set of basic objects, the *entities*, a set of collections, the *classes*, and a set of ways in which classes and entities can be related to one another, the *relations*. Such a formal, explicit representation of knowledge is called an *ontology*. An ontology can be created manually or learned automatically from data, but either approach is challenging. Assuming the existence of a suitable ontology, another challenge is mapping from natural language text to such a structured representation. For relations, this mapping is called *relation extraction*. Most successful approaches use supervised machine learning to generate extractors from labeled training data.

Unfortunately, these methods require a very large amount of human effort. Table 1 summarizes major evaluations of relation-extraction research. For each evaluation, only a relatively small number of relations were extracted, but in each case more than 100,000 annotated training words were provided, and in each case participants were given about 12 months of time for developing an extractor. Moreover, annotators and participants consisted of teams of experts working jointly.

Why does it require so much effort to create an extractor? One factor is annotation time. Supervised approaches require hundreds or thousands of training examples per relation. To reduce label-

Evaluation	# Relations	Training set	Development time
ACE 2004	51	300K annotated words	12 months
MR IC 2010	15	115K annotated words	12 months
MR KBP 2011	16	118K annotated words	12 months

Table 1.1: Evaluations of relation extraction. In each evaluation, only a small number of relations were extracted, but a large amount of human effort was necessary for annotation and development.

ing effort, unsupervised and weakly supervised approaches have been proposed [120, 112, 128], but the quality of extractions of such approaches tends to be low, both in recall and precision. The second factor is development time. One often tries to iteratively adapt an extraction model, for example by manually adding linguistic or domain-specific rules or features. However, that requires not only a deep understanding of language and domain, but also of the system itself and its algorithms. Moreover, it takes time to analyze errors, identify and select rules or features, and re-compute results.

Our goal in this work is to enable users to create a quality extractor in only *one hour*, using *no* previously annotated training data. We will do this by creating a novel tool that allows a user to build relation extractors extremely quickly. An important property of this tool will be that it enables most performance gains early and with very little user effort, and then allows a user to analyze and refine the extractor as needed. There are two key ideas that enable this tool and they can be summarized with the following thesis statement:

> We can greatly reduce the amount of human effort necessary to create relation extractors by leveraging (1) a richer set of interactions and (2) more accurate weakly supervised learning.

By supporting a richer set of interactions, we are not only going to make it quick and easy to provide new annotations, but also to integrate different components and data, perform analyses, as well as discover and choose new rules or refine existing ones. Some of these new interactions that we enable are actually quite simple for the user, but they incorporate new machine learning techniques based on weak supervision from a database.

In the rest of this introduction we first provide a broad overview of the problem of converting text into a knowledge base to provide the context for this dissertation. We then present our vision of

a novel tool for creating extractors and describe the interactions it enables. Finally, we discuss the challenges for each interaction and the contributions of this dissertation.

1.1 Converting Natural Language Text into Knowledge

There exists an enormous amount of work on extracting knowledge from natural language text. To better explain how our work fits in, we start with a simple example.

1.1.1 A motivating example

We assume that we are given the following snippet of text:

Tim Cook, the CEO of Apple, gave a keynote today. He announced the iPad product.

One goal of extraction is to be able to use such text to automatically answer questions such as:

What product did Cook announce?

In this case the system should simply return iPad . To enable this result, we can try to first convert the original text into a set of formal statements which we call our *knowledge-base*. A popular approach is to represent these statements as triples.

⟨ isCEOof, Cook0, Apple0 ⟩
⟨ talkHasFormat, t0, Apple0 ⟩
⟨ talkIsPresentedBy, t0, Cook0 ⟩
⟨ talkOnDay, t0, 06/21/2012 ⟩
⟨ announced, Cook0, iPad0 ⟩
⟨ isA, iPad0, product ⟩

Each triple contains one type of predefined binary relation such as isCEOof or talkHasFormat as well as two arguments, each refering to a unique entity. Here, the entity Apple0 represents Apple, the company. Another entity, Apple1, might represent the concept of the fruit. Since the relation and arguments are unambiguous, we say that they are *canonicalized*.

We can also convert our question into triple notation, so that we can query a database:

⟨ announced, Cook0, ?x ⟩ ⟨ isA, ?x, product ⟩

Here, ?x represents a variable that ranges over entities and whose value we would like to compute. For this knowledge-base, our query is satisfied for the solution set ?x$\in$ {iPad}.

In practice, not every example is that easy. The following sections highlight some of the challenges.

1.1.2 Ontologies

In our example we assumed that we are given a set of predefined relations such as isCEOof or talkHasFormat. How many are there? 100? 1000? 10000? Which individuals exist and how can they relate to one another? To answer such questions we use an ontology. An ontology gives us a shared vocabulary that defines the concepts and relations in a domain.

Researchers have tried building large ontologies that allow us to model the information encoded in a wide range of text. Unfortunately, this task is very difficult: There typically exist many different conceptualizations, each capturing different aspects, and merging them quickly leads to inconsistencies. Even a seemingly simple relation like is-A becomes tricky, since it is unclear what the determining features are. For example, when is a sandwich a hamburger and when is it just a sandwich? Does a car need to have exactly four wheels? Sometimes such decisions are based on science, other times on social consensus or tradition. Simply put, there exists no single ontology that we can use to organize all information. Instead of attempting to create an ontology with very broad coverage, a more promising approach is to allow many different ontologies to co-exist. Each ontology can then provide a different perspective on the information encoded in text, and might be more suitable for a different type of end-user application. Sometimes different ontologies overlap and an application might need to use several ontologies simultaneously. In such cases it is convenient to have links between them, even if the ontologies are not deeply integrated.

Ontologies can be created manually or learned automatically from data in an unsupervised fashion. Since manual ontology engineering can be difficult and time-consuming, learning-based approaches are attractive. These often perform some kind of clustering of synonymic expressions, and are based on a wide variety techniques, including topic models [131], spectral algorithms [31], Markov networks [120, 121, 88], neural networks [36, 15, 146], and open information extraction [11, 56, 58, 100, 88].

While being able to create ontologies without human input is a great advantage, there are also potential problems. For example, the learned conceptualization may not match a user's expectations. Applications which create structured views over the encoded information may then show inconsistencies and behave unexpectedly. For example, should a browsing interface include fictional characters? Should it include information that is only believed to be true by some people? Consistency is an important usability principle [142], and could be enforced by manually created ontologies. Furthermore, it may be difficult for a user to understand what has been learned. As we will see, an important part of this thesis will be work aimed at enabling users to interact with information extraction systems so that they can adapt them to their needs. A manually-created ontology makes it easier for users to manipulate the system, for example by providing extraction rules, defining relations, or adding world knowledge. Finally, we note that there already exists a set of manually created ontologies which have been found to be useful and which we would thus like to leverage. In particular, we will make significant use of Freebase and Wikipedia infobox ontologies.

In cases where there does not exist an appropriate ontology, we would like to have a simple, lightweight method for users to create one. Since automatic techniques can reduce user effort, the best approach may be a combination of an automatic and a manual approach. In this case, the manual component could allow users to define new concepts and relations or refine existing ones, and the automatic component could make suggestions based on statistical regularities in text and the user's feedback so far. In this thesis we leave combinations of manual and automatic ontology construction techniques as future work. Since we are interested in user interaction with ontology-based relation extraction systems, we generally assume manually created ontologies — both existing ones and ones created on-the-fly.

1.1.3 Relation Extraction

Given an ontology and a snippet of text, the key challenge addressed in this thesis is *relation extraction* — filling the relations stated in the text into the ontology. This task is surprisingly difficult, because there exist so many ways of expressing the same relations in text, and very often the same text could be ambiguous. Researchers have therefore tried a wide variety of techniques, often varying the precise definition of the task. We give a high-level overview of a broad range of work on

Input	Output				System
	Relation	Arg1	Arg2	Example	
text, KB	canon.	canon.[1]	canon.[1]	⟨isCeoOf, Cook0, Apple0⟩	SOFIE [149], PROSPERA [113]
text, seed KB, user	canon.	string	string	⟨isCeoOf, "Cook", "Apple"⟩	NELL [23, 24]
text, KB	canon.	string	string	⟨isCeoOf, "Cook", "Apple"⟩	**MultiR** [80], Riedel10 [127], MIML [150], Krishn12 [89]
text, infobox	canon.	canon.	string	⟨isCeoOf, Cook0, "Apple"⟩	**Luchs** [81], Kylin [168]
text	clustered	clustered	clustered	⟨id27, id19, id55⟩	USP [120], OntoUSP [121]
text, infobox	string	string	string	⟨", the CEO of", "Cook", "Apple"⟩	WOE [170]
text	string	string	string	⟨", the CEO of", "Cook", "Apple"⟩	TextRunner [11], ReVerb [58], DIRT [99]

Figure 1.1: Inputs and outputs of systems extracting binary relations. Many systems only partially canonicalize relations and arguments.

relation extraction and its challenges in this section, and then provide formal task definitions when we discuss our proposed systems in chapters 2 and 3.

The output of a relation extraction system is a set of identified relation instances. Our focus will be on identifying instances of *binary* relations and finding their argument mentions in text. We note that there also exist approaches for extracting n-ary relations [107, 71], but these will not be discussed in this thesis. Each relation instance is associated with a canonical relation in the ontology. As a first step, one does not always canonicalize the arguments to entities in the ontology, but sometimes represents them as the strings used in the text. For some applications, the strings themselves can be useful [83], for others they can be linked to the ontology using specialized algorithms [39]. The work to date on relation extraction has varied along these degrees of canonicalization of relations and arguments, as shown in Figure 1.1. In this figure, MULTIR and LUCHS which are described in this dissertation are highlighted in bold.

The input can be just text, for example if an ontology is learned at the same time. Often, additional inputs are supplied. These can be manually written patterns, annotations of relations in text, coupling constraints in an existing ontology, a populated knowledge-base of known facts, lexicons

[1] System only extracts relations between known entities.

of related words and paraphrases, and many other inputs. Additional inputs can improve the quality of the output, but sometimes make the task more difficult to model.

To make the problem more manageable, it is typical to break it down into subproblems which can be solved separately. Subproblems include tokenization, part-of-speech tagging, syntactic parsing, named-entity recognition, named-entity linking, coreference resolution, argument identification, and more. Since many of these subproblems are very challenging by themselves, it is common to use existing components where available.

Depending on inputs, outputs and the existing components that are used, the problem can be modeled in different ways. Traditionally, relation extraction was often seen as a *pattern matching* task, where patterns were expressed as regular expressions over lexical items or part-of-speech tags [7, 130, 76, 144]. Another way is to consider it as a *sequence tagging* task, where the sequence consists of the lexical items or part-of-speech tags of the sentence, and the tags are argument labels [11, 168, 81, 137]. Yet another approach is to see it as a *classification* task. Often, potential arguments are identified first, for example using a part-of-speech tagger, named-entity tagger, or syntactic parser. Pairs of arguments are then classified to a relation label or a distinguished label for no relation [86, 21, 112, 80]. It is also possible to first identify trigger words that indicate a relation, and then classify nearby noun phrases to the relation's arguments [71]. Another approach is to see relation extraction as a *parsing* task. Here, a semantic parse is computed which decomposes the sentence into both syntactic and semantic units. The semantic parse makes the relations explicit [89, 173]. Finally, approaches that do not use a manually created ontology as input often treat relation extraction as a *clustering* task, where the goal is to group semantically similar phrases [120, 121].

What makes relation extraction so difficult is that there exist so many ways in which the same relation could be expressed in text. Also, the same phrases can sometimes mean different things. Consider the following examples:

Tim Cook, the CEO of Apple, ...
The CEO of Apple, Tim Cook, ...
Cook who is CEO of Apple, ...
Tim is the chief executive of Apple.

Apple ‘s chief executive, Timothy Cook, . . .

Cook who heads Apple . . .

John heads home.

John, the head of the church, . . .

The first six sentences contain variations of the isCEOof relation, whereas the last three sentences all contain the same verb to head but with different meanings.

To understand how well a relation extraction system can handle these challenges, one typically measures its performance in terms of *precision* and *recall*,

$$\text{Precision} = \frac{tp}{tp + fp},$$

$$\text{Recall} = \frac{tp}{tp + fn}.$$

Here, tp (true positives) is the number of relation instances that are identified and that are truly expressed in the text, fp (false positives) the number that are identified but that are not actually expressed in the text, and fn (false negatives) the number that are expressed in the text but missed. Intuitively, high recall means that the system understands many variations of the same relations, and high precision means that it does not confuse different relations in its output. If a single metric is desired, one can combine precision and recall to their harmonic mean, which is called *F-measure*,

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Many techniques have been proposed to increase precision and recall. To increase precision, one often tries to add additional constraints. For example, type constraints ensure that the arguments of a relation match a predefined type. Types can be inferred by an independent model such as a named entity tagger. Other constraints might ensure that there is not more than one extraction for a functional relation such as birth date. Furthermore, some relations are mutually exclusive. For example, an entity cannot be an academic field and a chemical at the same time. Ideally, one would like to consider multiple constraints and the potential predictions of different components all at once, to find the best overall configuration. Such joint-inference can deliver even more precise

results [119].

Other techniques have been proposed to target recall. A popular idea is to leverage a large amount of unlabeled text, for example using bootstrapping [7, 99]. Another approach is to use other resources such as WordNet [110, 95], sets of lexicons mined from Web lists [81, 165], or clusters of words mined from text [18]. One can also use unsupervised techniques to identify and cluster relational phrases in a massive text corpus such as the Web [11]. Combining relation extraction with other models such as coreference resolution can help [132], and joint-inference may increase recall, too.

To reach adequate precision and recall, however, most successful relation extraction systems also rely on manually created annotations of relations in text and use these for supervised learning of a relation extraction model. Unfortunately, this typically requires hundreds or even thousands of annotated examples per relation, and is thus too expensive for many practical applications. Instead of relying on manually created annotations, more and more researchers are therefore trying to leverage weaker signals that can be obtained more cheaply. One such signal are matches of extractions to an existing database [112]. Often, however, the quality of a weakly supervised extractor is unsatisfactory or it might even be impossible to learn a weakly supervised extractor.

Another important problem is that developing an extractor is often time-intensive and difficult for users without the necessary skills, even if there exist annotations that can be used for training. In practice, users often need to create, evaluate, and refine extraction patterns. Frequently, they directly manipulate an algorithm or statistical model, or they try to incorporate additional resources such as dictionaries or databases. Such actions are clearly impossible without a deep understanding of the subject, and it may be difficult to do them quickly. Furthermore, it often takes time to re-compute an extractor, to analyze its performance, and to discover new ways of improving it.

1.2 Envisioning a Novel Tool for Relation Extraction

In this work, our goal is to enable users to create a quality extractor in only one hour, using no previously annotated training data. We assume that our users have some understanding of linguistics and logic. For example, we expect that they understand that there exist syntactic dependencies between words in a sentence, and that they know how to read logical expressions. We will not

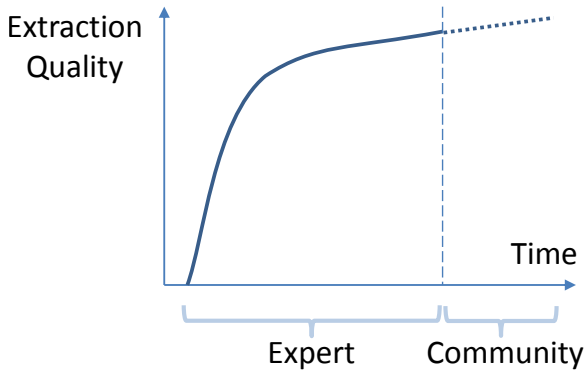
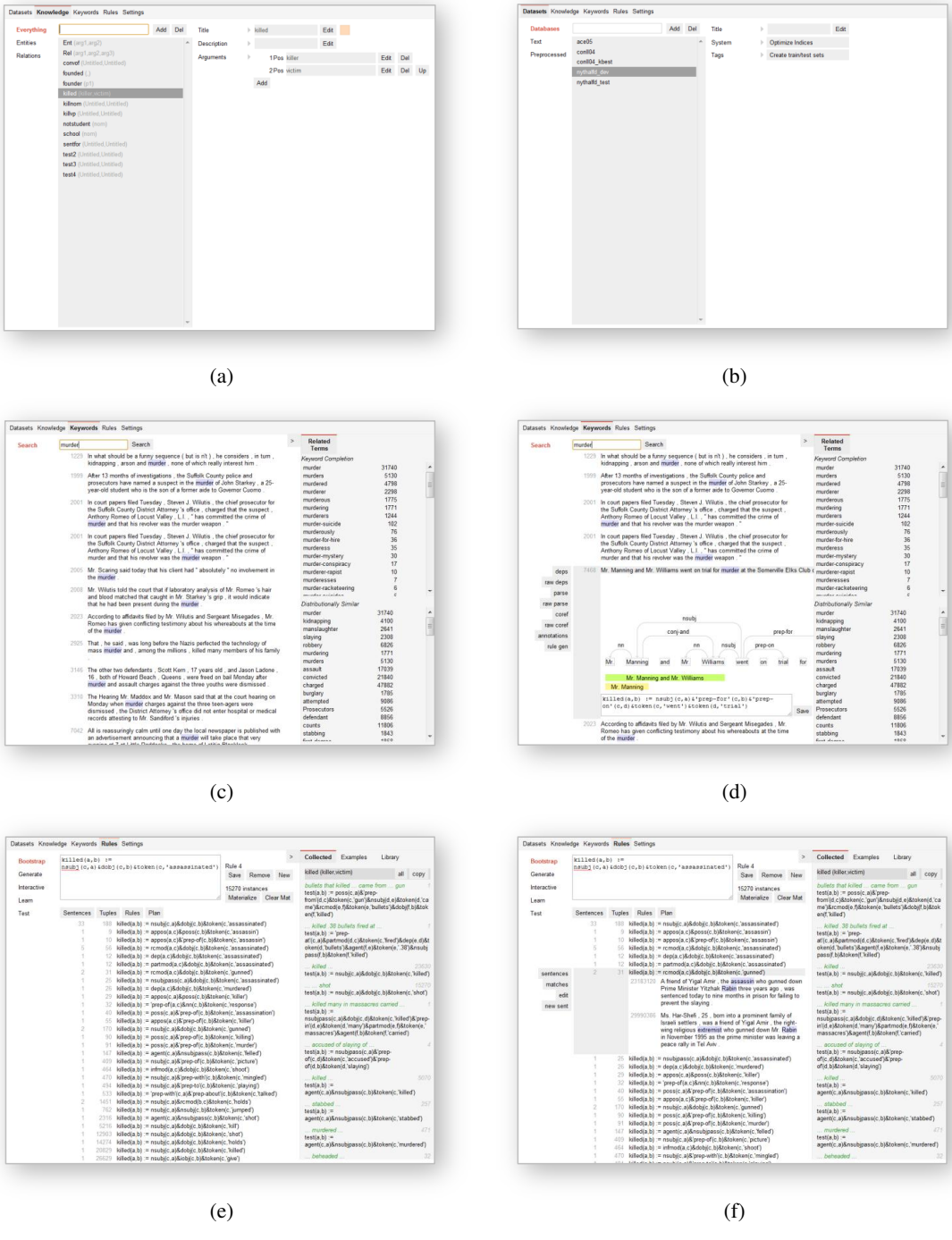


Figure 1.2: The interactions with the tool are designed to enable most gains in extraction performance already early on. After an expert has created an initial extractor, one can leverage an online community to continue to improve it.

assume, however, that they have knowledge in machine learning, algorithms, or how an extraction system can be implemented.

The key question we ask in this dissertation is how can we enable such users to create a quality extractor so quickly? For us, the solution is the design of a new tool for creating extractors, and this design focuses on the interaction between user and tool, trying to make it easy and efficient. Users can interact with the tool in multiple ways by manipulating data and easy-to-understand models; at no point, however, are they expected to develop or refine an algorithm. The different interactions that the tool offers are not only very easy for users, they are also designed to enable most gains in extraction performance as quickly as possible (Figure 1.2). Nonetheless, they can refine an extractor whenever they wish to do so.

The interaction between user and tool is best explained by looking at an illustrative example. We assume that our user, Anna, is interested in creating an extractor for the killed(killer,victim) relation. After loading the tool in a Web browser, she performs the following steps:



(a) (b) (c) (d) (e) (f)

Figure 1.3: Selected interactions with our envisioned tool. Users interact in multiple ways as described in section 1.2.

- Anna starts by defining the killed(killer,victim) relation in the tool (Figure 1.2a). She sets its arguments to be word positions in a sentence containing the relation.

- She then chooses a development dataset, in this case the NYTimes corpus with 45M newswire sentences (Figure 1.2b). She will later use this dataset to build and analyze her extractor.

- Anna finds an existing database of the world's most famous murderers and their victims, and imports it into the tool. Using weak supervision from that database, the system automatically bootstraps an extractor.

- Curious to see how well this extractor does, Anna checks its behavior on a sample of sentences. She searches for sentences containing keywords such as 'murder', 'gun', or 'Capone' (Figure 1.2c).

- She notices a few instances that were missed by the extractor, and checks the syntactic structure of the relevant sentences. With a few mouse clicks, she is able to generate a new rule which captures the instances that were missed (Figure 1.2d).

- At that moment, the tool displays a ranked set of additional automatically generated rule suggestions, which have been obtained by bootstrapping from existing rules and tuples (Figure 1.2e). Anna checks the behavior of these rules and notices that they cover additional cases she had not considered (Figure 1.2f). She adds five of them to her extractor.

- After a few more minutes of inspecting sentences and developing rules, Anna is satisfied with her extractor. However, she would like it to keep improving without her input. Using the tool, she thus launches a campaign that leverages an online community to validate extractions.

- As the community provides extraction validations, the tool uses these to automatically refine the extractor. Over time, it is becoming increasingly more accurate.

In this dissertation we would like to study the interesting research questions which our envisioned tool presents. The key to our tool is a richer set of interactions, and we believe that several of

these interactions warrant further research investigation. We thus select three interactions which we find particularly interesting from a research perspective, and study them in detail, identifying their challenges and developing novel solutions:

Providing databases of relation instances. This first interaction is a very simple one to the user, who merely selects columns in an existing database. The heavy-lifting, however, is performed by the tool which tries to automatically bootstrap an extractor based on the structured content of the database. With this interaction, it is sometimes possible to create an extractor with just the push of a button.

Creating and refining extraction rules. Unfortunately, it is not always possible to bootstrap an extractor from a database, so we need a fallback solution. For us, this fallback solution is to make manual rule writing quick and easy. Although that requires more than the push of a button, we can try to minimize user effort, for example by providing feedback instantly and automatically suggesting new rules.

Community-based validation of extractions. Although one can get very far in creating extractors in just one hour, it would be great if an extractor could later be improved, without additional input by an expert. To make this possible we are interested in leveraging an online community of non-expert users by motivating these to contribute through simple interactions for validating extractions.

Each of these types of interaction presents different research problems. In the following sections, we consider these interactions in more detail, giving an overview of existing work and discussing the important challenges.

1.2.1 Providing databases of relation instances

When users provide a database of relation instances, one can try to automatically bootstrap an extractor. One such approach is *knowledge-based weak supervision*, a technique which heuristically matches the contents of the database to the text in order to automatically create training data for learning an extractor. Consider the following example sentences and database with company CEOs:

1. [Tim Cook] is the chief executive of [Apple].
2. [Tim Cook] 's comments reveal little about [Apple]'s future.
3. [Fernando Aguirre], CEO of [Chiquita], extolled healthy workplace practices on Friday.
4. Stephen Colbert is scared of Amazon founder and CEO Jeff Bezos.
5. [Chiquita], the banana company, recently hired [Fernando Aguirre].
6. [Jørgen Knudstorp] came to family-owned [Lego] as an outsider in 2001.

isCEOof

Tim Cook	Apple
Fernando Aguirre	Chiquita
Jørgen Knudstorp	Lego
Steve Ballmer	Microsoft

By taking the arguments of tuples in the database and matching them to sentences, one obtains new annotations. These annotations can then be used to train a relation extraction model just like one would train a supervised relation extraction model. Specifically, we define a 'mention' as a pair of named entities appearing in a sentence. If a mention is matched to a tuple in the database we call it an 'aligned mention' and give it a positive label, otherwise a negative label. From this data, we can then learn a classifier. So, instead of manually annotated sentences, one only needs a large database containing relation instances and a large amount of unlabeled text that these relation instances can be matched to.

However, this technique only works if we make certain assumptions. In our simple example, we assumed that (1) we have a database table that contains tuples of the relation we are interested in, (2) that all such tuples expressed in our text are contained in our database, and (3) that any sentence that contains a string match for every column of a tuple in the table expresses that relation.

Certainly, these assumptions are not always true. Although we did have a database with company CEOs in our example, it did not contain Amazon's CEO Jeff Bezos. Therefore, our heuristic failed

to annotate sentence four. Furthermore, only the first, the third, and the fourth sentence express the isCEOof relation, while the remaining sentences express other relations for the same pairs of individuals. This raises the following two questions: How likely is it that our assumptions are violated? What should we do if our assumptions are violated?

It is clear that we can only apply this technique for relations for which there exist comprehensive databases. Today, we have large knowledge bases like Freebase, YAGO, and Factual, that contain a wide variety of relations. These relations include things like birth places of celebrities, and the leagues a baseball player played in. It is important to note, however, that although these knowledge bases now contain billions of facts, they still only cover a small percentage of the knowledge conveyed through text.

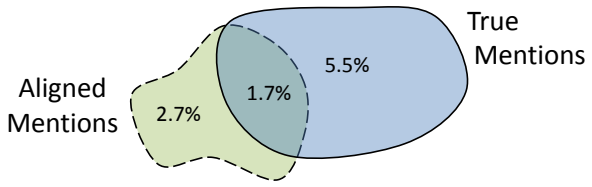


Figure 1.4: Comparison of manual annotations and weakly-supervised annotations. 2.7% of pairs of named entity mentions in 45M sentences of the New York Times can be matched to instances of the 50 most popular relations in Freebase, but only 1.7% truly express those relations.

Moreover, in cases where we have a large number of relation instances, the matches can be surprisingly noisy. To see this, we performed an experiment where we matched all instances of the 50 most popular Freebase relations to all sentences of news articles published in the New York Times between 01/1987 and 06/2007. We then sampled 2000 random sentences from this news corpus, and manually checked if they contained mentions of the 50 relations. As shown in Figure 1.4, we found that the automatic matching returned 1.6 times as many false positives as true positives, and 3.2 times as many false negatives as true positives. In other words, the majority of times that a sentence expressed a relation of interest, it did not match, and more than 1/3 of the returned matches were incorrect.

When used directly to train an extractor, such noisy annotations tend to confuse the learning algorithm, and the quality of the extractor deteriorates. To mitigate this problem, one approach is to explicitly model the uncertainty in the annotations. The task then becomes to simultaneously pre-

dict both the true annotations and the relational extractors. This can be done by casting the problem into the framework of multi-instance learning [20, 127, 80]. With multi-instance learning, one no longer assumes that *every* sentence matching a relation instance in the database expresses that relation. Instead, one merely assumes that *at least one* sentence matching a relation instance expresses that relation. This weaker assumption can make the learning procedure more robust, but there are additional challenges: Existing approaches are too slow to scale to large datasets. Furthermore, a large number of relations in Freebase overlap. For example, both isCEOof(Cook,Apple) and isShareholderOf(Cook,Apple) are true. Existing approaches do not model such overlaps, decreasing the quality of their results. In chapter 2 we therefore propose a novel method which addresses these problems. Our proposed method performs multi-instance learning, scales to large datasets, and successfully handles overlapping relations. Its main idea is to learn an extractor that uses only sentence-level features, but to leverage the aggregate information to model weak supervision during training.

A different approach to handle noise is to be more selective about which text and database to align. Only when it is known beforehand that that the text and database likely contain the same relations, one performs the matching. An example where this approach can be applied to is matching facts in Wikipedia infoboxes to their articles. Wikipedia *infoboxes* are short summaries containing the main facts of a Wikipedia article in structured form. Since many relations are contained in both the article and the infobox, the annotations are more likely to correspond to true mentions of the relations. It is possible to further increase accuracy by only considering the top sentences in an article. This is because important facts such as those in the infobox tend to be mentioned early in an article. There are several challenges, however: If the goal is to learn extractors that work on general text, one has to ensure not to overfit to Wikipedia articles. In work which is not part of this dissertation, we showed that such extractors can indeed perform well on the broader Web [167]. A bigger problem is that there only exists a small number of training examples that can be obtained this way for each relation, which severely limits recall. In chapter 3 we propose a novel method that can learn an extractor from such sparse data. This method uses a very large number of lists appearing on Web pages to automatically create a small set of lexicons specific to each relation extractor. These lexicons are then used to bias the extractor, so that it can more accurately learn from sparse training data.

1.2.2 Creating and refining extraction rules

While knowledge-based weak supervision holds great promise, it cannot always be applied successfully. Large-scale knowledge bases such as Freebase, YAGO, or Factual only cover a relatively small set of *factual* knowledge, while the vast majority of information expressed in natural language text does not exist in structured form. For example, information about opinion, dialogue, or cause is missing entirely from these sources. Certainly, the volume of information available in knowledge-bases is growing, making learning with knowledge-based weak supervision increasingly more applicable, but the gap will likely remain large. Furthermore, even when the relations of interest are available in structured form, uncertainty in the alignment of relations to text may lead to unsatisfactory results. Although different techniques alleviate this problem, weakly supervised extractors may not be competitive with supervised ones. So what can one do if weakly supervised (and unsupervised) techniques cannot be applied?

In such cases we will need to collect additional human feedback. Recall, however, that such feedback can be extremely expensive, with supervised extractors often requiring hundreds or even thousands of training examples per relation. So, how can we make better use of human effort? Rather than collecting labeled data, we could try to collect extraction patterns. We could also enable users to directly manipulate an algorithm or statistical model. However, we have to be careful: While such approaches could reduce the amount of human effort, they may require a deeper understanding of the system and only experts may be able to perform them. It is thus important to take into account the skills of users when designing new interactions.

Existing research has proposed solutions based on active learning [51, 138], learning with constraints [27, 30, 13], and new modeling languages [48]. Although some of these approaches require expert knowledge, they make it easier for users to provide feedback through manipulating data or easy-to-understand models, rather than tweaking an algorithm. We, too, are interested in following this principle. More specifically, we would like to enable experts to create rule-based extractors very quickly. Rules are very important, for both hand-built and learned systems. For example, a typical rule for relation extraction such as

$$\begin{aligned} \texttt{r}(\texttt{a},\texttt{b}) \Leftarrow\ & \texttt{PER}(\texttt{a}) \wedge \texttt{nsubj}(\texttt{c},\texttt{a}) \\ & \wedge \texttt{token}(\texttt{c}, \text{`}\texttt{born}\text{'}) \\ & \wedge \texttt{prep-in}(\texttt{c},\texttt{b}) \\ & \wedge \texttt{LOC}(\texttt{b}) \end{aligned}$$

combines lexical information (here token 'born'), entity type information (here types PER and LOC), and syntactic information (here dependencies nsubj and prep-in). Such rules can be used in a variety of ways, for example as extraction patterns, features, or constraints, and the choice of rules has often a huge impact on performance. With the right rules, an extraction system might work very well. In fact, many state-of-the-art systems for certain natural language processing tasks are purely based on simple, deterministic rules. Since it is also possible to learn rules automatically, they may be a good interface for more tightly integrating approaches based on human feedback and statistical learning in the future.

Our goal in this work is to enable experts to write quality rules extremely quickly, but there are several challenges. For example, when developing an extractor one would often like to combine existing components that are specialized for different subtasks such as named entity recognition, named entity linking, or coreference resolution. Such components, however, are usually not easily pluggable, since there are no standardized interfaces. Furthermore, one cannot easily add constraints that connect the output of multiple such components. Sometimes, one would like to refine one component and analyze the impact on the complete system, but without tool support this can be a tedious task. There is no lightweight way for manipulating one part of the system, and then seeing the results on a large dataset, instantly. There is also no simple way for discovering which new rule might improve the system. In chapter 4 we therefore propose a novel interactive system that tackles these problems, and evaluate its effectiveness. Our system encodes all outputs and inputs of different components as logical predicates, and then allows users to write logical rules to define new predicates. All predicates are also indexed in multiple ways, so that the effect of any rule can be seen instantly, even on large datasets. Finally, the system enables boostrapping to help users discover new rules.

1.2.3 Community-based validation of extractions

While our previously-described methods allow experts to create acceptable extractors in just one hour, one often wishes to improve an extractor afterwards, without the need for additional expert input. We thus turn our attention to interactions for non-experts.

While interactions for experts are often designed with the goal that human effort is used efficiently, interactions for non-experts are often more concerned with motivating large numbers of users to participate and with ensuring the quality of feedback. Furthermore, in contrast to approaches for experts, the types of input for the latter tend to be more constrained. Existing work has largely focused on paid crowdsourcing platforms such as Mechanical Turk [145] to collect feedback, but is it necessary to pay workers for feedback when there exist large communities that create content like Wikipedia for free? In this work we are interested in leveraging such online communities of users who are not experts in relation extraction in order to improve a relation extractor.

In particular, we would like to collect feedback from users visiting a Wikipedia article. There are many challenges and open questions. For example, these users are not only non-experts, they may also have little motivation to help us create an extractor. When they visit a Wikipedia article, they are engaged with another primary task (i.e. the task that brought them to the Wikipedia article in the first place). So, how can we collect valuable feedback from such users? Our goal is to entice users to spontaneously choose to contribute by making the opportunity to contribute clearly visible, and by developing interactions that are both quick and easy. But how much can we increase the contribution rate, without being considered too intrusive? More importantly, will the feedback we collect that way improve our extractors? How can we even test this? Can we synergistically combine feedback from the community and learning an extractor such that one amplifies the other? In chapter 5 we present a series of experiments yielding preliminary answers to these questions. In particular, we propose three interfaces for validating extractions as a non-primary task, and measure their effectiveness by directing users to these interfaces using search advertising. We show that we can significantly increase the contribution rate and that the obtained contributions significantly improve our extractors.

1.3 Contributions

The previous sections outlined the difficulties that we need to overcome if we want to make our envisioned tool a reality. In short, we need to increase precision and recall of weakly supervised extractors, we need to make expert feedback more efficient, and we need to be able to leverage online communities more effectively.

The goal of this dissertation is to provide answers to these challenges. Since they have little overlap, it is useful to study them separately. This dissertation thus presents the design, implementation, and evaluation of several novel systems for enabling scalable, ontology-based relation extraction.

MULTIR**: Modeling Overlapping Relations in a Multi-Instance Learning-Based Approach to Weakly Supervised Relation Extraction.** Existing systems for learning relational extractors with knowledge-based weak supervision assume that relations are disjoint. For example, they assume that both isCEOof(Cook, Apple) and isShareholderOf(Cook, Apple) cannot be true at the same time. To *increase precision* we introduce MULTIR, a system based on a novel statistical model that uses multi-instance learning to combat the noise in the training data and also handles overlapping relations. The system combines a sentence-level extraction model with a simple, corpus-level component for aggregating the individual facts. Experiments show that the approach runs quickly and yields surprising gains in accuracy, at both the aggregate and sentence level. We present MULTIR in chapter 2.

LUCHS**: Learning Extractors from Sparse Training Data using Web-Scale Semi-Supervised Lexicon Generation.** A challenge of learning extractors with knowledge-based weak supervision is that there are often only a small number of matches per relation. This means that only few heuristic training examples can be generated, and as a consequence learned extractors often suffer from low recall. In response, we developed LUCHS, a system that automatically induces features using semi-supervised lexicon learning in order to *increase recall* of knowledge-based weak supervision. LUCHS starts by harvesting HTML lists from a 5B document Web crawl. When learning an extractor for relation R, LUCHS extracts seed phrases from R's training data and uses a semi-supervised learning algorithm to create several relation-specific lexicons at different points on a precision-recall

spectrum. These lexicons form Boolean features which enable it to learn 5025 relational extractors — more than an order of magnitude greater than any previous approach — with an average F-measure of 61%. We discuss LUCHS in chapter 3.

INSTAREAD**: Creating Relation Extractors Interactively.** Knowledge-based weak supervision is often not applicable, either because the quality of the learned extractors is unsatisfactory or because there does not exist a database that can be used. We therefore introduce INSTAREAD, a system that enables expert users to create high-quality rule-based extractors in a limited amount of time, using no labeled training data or database. The key to this technique is a combination of an expressive rule language based on first-order logic, an interface accelerating rule authoring by recommending rules and showing relevant information, and use of a database management system to enable instant rule evaluation. Experiments show that the system executes most rules in less than 74ms on a corpus of 22M sentences, and that 55 minutes of development time yield an F-measure of 58%. INSTAREAD is presented in chaper 4.

SMARTWIKI**: Synergistic Pairing of Relation Extraction and Community Content Creation.** When learning with knowledge-based weak supervision delivers extractors of unsatisfactory quality, it may be possible to improve quality without the need for expert feedback. Instead, we seek to leverage input from an online community. We present SMARTWIKI, a system that synergistically pairs a relation extraction model with community content creation. SMARTWIKI presents the verification of relation extractions as a non-primary task in the context of Wikipedia articles. We demonstrate the proposed synergy by analyzing a SMARTWIKI deployment from two perspectives: First, we show it accelerates community content creation by using relation extraction to significantly increase the likelihood that a person visiting a Wikipedia article will spontaneously choose to help improve the article's infobox, and, second, we show it accelerates relation extraction by using contributions collected from people interacting with SmartWiki to improve relation extraction performance. Chapter 5 details SMARTWIKI's design.

Overall Contribution The ability to automatically convert natural language text into a knowledge base opens up great new opportunities, but existing approaches are either not reliable or demand a

prohibitive amount of human effort. On a high-level, this dissertation argues that to solve this problem we need to be looking at richer forms of human feedback. These richer forms of feedback must be carefully designed with attention to the cost, benefits, and skills of each user. Expert users may be able to provide a wider range of types of feedback than non-expert users. Users who do not benefit from providing feedback may be less willing to do so, but there may be a larger number of them. Independent of each user's skills, we would like to ensure that their effort is used efficiently — yielding large improvements in extraction performance with little human effort.

This dissertation further presents the vision of an integrated tool for creating extractors which provides the proposed interactions with its users. Some of these interactions are based on more accurate weakly supervised learning, and so this dissertation studies novel solutions to this problem. It proposes a new model for learning extractors with knowledge-based weak supervision, which significantly improves over the existing state-of-the-art by handling overlapping relations. It further shows that knowledge-based weak supervision often suffers from sparse data, and proposes an approach to increase recall by automatically inducing features using semi-supervised lexicon learning. Such learning-based approaches enable simple interactions, which can get a user very far in creating an extractor with almost negligible effort. When such approaches fail, however, other interactions are needed. This dissertation thus also proposes an approach for developing rules interactively, and shows that the combination of a simple rule language, instant rule evaluation, and automatic rule suggestions can yield high-quality extractors in less than one hour. Further, it shows how one can leverage an online community of non-expert users to improve an extractor. Novel interactions entice users involved in another task to spontaneously choose to contribute. Finally, a discussion compares the different types of interactions, describing the contexts in which they are applicable and useful. One key conclusion is that to be most effective we need a tight integration of different techniques.

In summary, the techniques proposed in this dissertation push the boundaries of research on large-scale relation extraction. Not only are they novel, they have also been tested on millions of documents, and integrated into several concrete end-user applications such as ones for completing Wikipedia infoboxes or performing interactive information extraction. All systems involve a combination of techniques from multiple computing fields – including natural language processing, databases, machine learning, and human-computer interaction. We believe that these properties are key to enabling practical large-scale relation extraction.

Chapter 2

MULTIR: WEAKLY SUPERVISED EXTRACTION OF OVERLAPPING RELATIONS

If a user is able to provide a database of relation instances, one can try to automatically bootstrap an extractor using weak supervision. While this does not always work, it often allows a user to create an extractor with just the push of a button. To make learning with weak supervision applicable in more situations, one challenge is to develop a technique that can combat the noisy training data that can come from heuristic labeling. Recently, researchers have developed multi-instance learning algorithms, but their models assume that relations are *disjoint* — for example they cannot extract the pair isCEOof(Cook, Apple) and isShareholderOf(Cook, Apple).

This chapter presents a novel approach for multi-instance learning with *overlapping* relations that combines a sentence-level extraction model with a simple, corpus-level component for aggregating the individual facts. We apply our model to learn extractors for NY Times text using weak supervision from Freebase. Experiments show that the approach runs quickly and yields surprising gains in accuracy, at both the aggregate and sentence level.

2.1 Introduction

Since creating the labeled training data necessary for supervised learning of relation extractors is too expensive, we are interested in leveraging other types of human feedback. One such type is providing databases of relation instances. With databases, it is often possible automatically bootstrap an extractor without the need for additional human input. In section 1.2.1 we introduced *knowledge-based weak supervision*, one such approach which creates training data by heuristically matching the contents of a database to corresponding text. Consider the following running example. Suppose that $r(e_1, e_2) = \texttt{isCEOof}(\texttt{Cook}, \texttt{Apple})$ is a ground tuple in the database and $s =$"Tim Cook is the Apple's current CEO." is a sentence containing synonyms for both $e_1 = \texttt{Jobs}$ and $e_2 = \texttt{Apple}$, then s may be a natural language expression of the fact that $r(e_1, e_2)$ holds and could

be a useful training example.

While weak supervision works well when the textual corpus is tightly aligned to the database contents (*e.g.*, matching Wikipedia infoboxes to associated articles [81]), Riedel *et al.* [128] observe that the heuristic leads to noisy data and poor extraction performance when the method is applied more broadly (*e.g.*, matching Freebase records to NY Times articles). To fix this problem they cast weak supervision as a form of multi-instance learning, assuming only that *at least one* of the sentences containing e_1 and e_2 are expressing $r(e_1, e_2)$, and their method yields a substantial improvement in extraction performance.

However, Riedel *et al.*'s model (like that of previous systems [112]) assumes that *relations do not overlap* — there cannot exist two facts $r(e_1, e_2)$ and $q(e_1, e_2)$ that are both true for any pair of entities, e_1 and e_2. Unfortunately, this assumption is often violated; for example both `isCEOof(Cook, Apple)` and `isShareholderOf(Cook, Apple)` are true. Indeed, 18.3% of the weak supervision facts in Freebase that match sentences in the NY Times 2007 corpus have overlapping relations.

This chapter presents MULTIR, a novel model of weak supervision that makes the following contributions:

- MULTIR introduces a probabilistic, graphical model of multi-instance learning which handles overlapping relations.
- MULTIR also produces accurate sentence-level predictions, decoding individual sentences as well as making corpus-level extractions.
- MULTIR is computationally tractable. Inference reduces to weighted set cover, for which it uses a greedy approximation with worst case running time $O(|\mathrm{R}| \cdot |\mathrm{S}|)$ where R is the set of possible relations and S is largest set of sentences for any entity pair. In practice, MULTIR runs very quickly.
- We present experiments showing that MULTIR outperforms a reimplementation of Riedel *et al.* [128]'s approach on both aggregate (corpus as a whole) and sentential extractions. Additional experiments characterize aspects of MULTIR's performance.

2.2 Weak Supervision from a Database

Given a corpus of text, we seek to extract facts about *entities*, such as the company `Apple` or the city `Boston`. A *ground fact* (or *relation instance*), is an expression $r(\mathbf{e})$ where r is a relation name, for example `isCEOof` or `isShareholderOf`, and $\mathbf{e} = e_1, \ldots, e_n$ is a list of entities.

An *entity mention* is a contiguous sequence of textual tokens denoting an entity. In this chapter we assume that there is an *oracle* which can identify all entity mentions in a corpus, but the oracle doesn't normalize or disambiguate these mentions. We use $e_i \in E$ to denote both an entity and its name (*i.e.*, the tokens in its mention).

A *relation mention* is a sequence of text (including one or more entity mentions) which states that some ground fact $r(\mathbf{e})$ is true. For example, "Steve Ballmer, CEO of Microsoft, spoke recently at CES." contains three entity mentions as well as a relation mention for `isCEOof(Steve Ballmer, Microsoft)`. We restrict our attention to binary relations. Furthermore, we assume that both entity mentions appear as noun phrases in a single sentence.

The task of *aggregate extraction* takes two inputs, Σ, a set of sentences comprising the corpus, and an extraction model; as output it should produce a set of ground facts, I, such that each fact $r(\mathbf{e}) \in I$ is expressed somewhere in the corpus.

Sentential extraction takes the same input and likewise produces I, but in addition it also produces a function, $\Gamma : I \rightarrow \mathcal{P}(\Sigma)$, which identifies, for each $r(\mathbf{e}) \in I$, the set of sentences in Σ that contain a mention describing $r(\mathbf{e})$. In general, the corpus-level extraction problem is easier, since it need only make aggregate predictions, perhaps using corpus-wide statistics. In contrast, sentence-level extraction must justify each extraction with *every* sentence which expresses the fact.

The *knowledge-based weakly supervised learning* problem takes as input (1) Σ, a training corpus, (2) E, a set of entities mentioned in that corpus, (3) R, a set of relation names, and (4), Δ, a set of ground facts of relations in R. As output the learner produces an extraction model.

2.3 Modeling Overlapping Relations

We define an undirected graphical model that allows joint reasoning about aggregate (corpus-level) and sentence-level extraction decisions. Figure 2.1(a) shows the model in plate form.

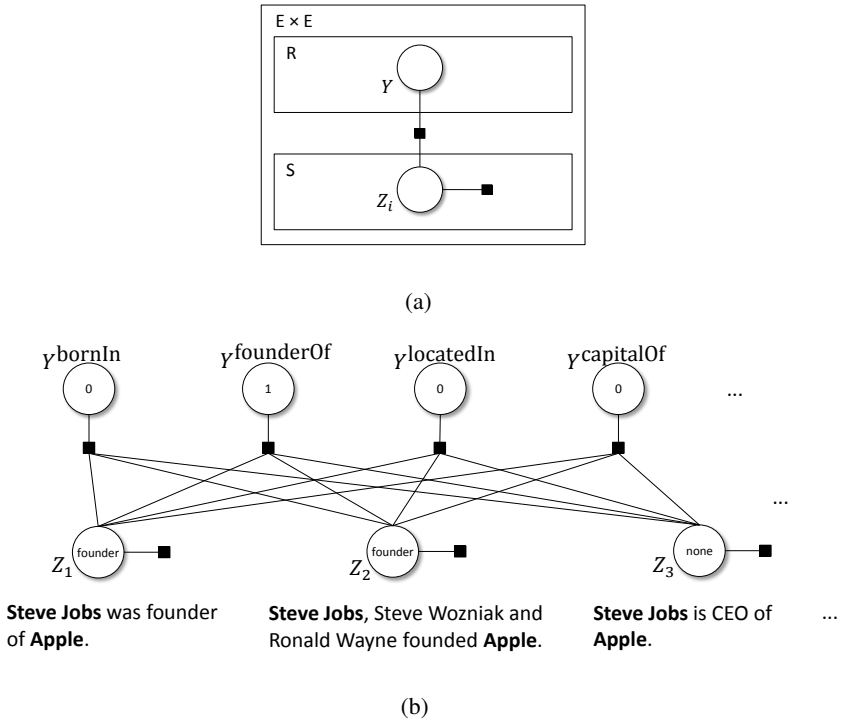


Figure 2.1: (a) Network structure depicted as plate model and (b) an example network instantiation for the pair of entities `Steve Jobs`, `Apple`.

2.3.1 Random Variables

There exists a connected component for each pair of entities $\mathbf{e} = (e_1, e_2) \in \text{E} \times \text{E}$ that models all of the extraction decisions for this pair. There is one Boolean output variable Y^r for each relation name $r \in \text{R}$, which represents whether the ground fact $r(\mathbf{e})$ is true. Including this set of binary random variables enables our model to extract overlapping relations.

Let $S_{(e_1,e_2)} \subset \Sigma$ be the set of sentences which contain mentions of both of the entities. For each sentence $x_i \in S_{(e_1,e_2)}$ there exists a latent variable Z_i which ranges over the relation names $r \in R$ and, importantly, also the distinct value `none`. Z_i should be assigned a value $r \in R$ only when x_i expresses the ground fact $r(\mathbf{e})$, thereby modeling sentence-level extraction.

Figure 2.1(b) shows an example instantiation of the model with four relation names and three sentences.

2.3.2 A Joint, Conditional Extraction Model

We use a conditional probability model that defines a joint distribution over all of the extraction random variables defined above. The model is undirected and includes repeated factors for making sentence level predictions as well as globals factors for aggregating these choices.

For each entity pair $\mathbf{e} = (e_1, e_2)$, define $\mathbf{x}$ to be a vector concatenating the individual sentences $x_i \in S_{(e_1,e_2)}$, $\mathbf{Y}$ to be vector of binary Y^r random variables, one for each $r \in R$, and $\mathbf{Z}$ to be the vector of Z_i variables, one for each sentence x_i. Our conditional extraction model is defined as follows:

$$p(\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}|\mathbf{x}; \theta) \stackrel{\text{def}}{=} \frac{1}{Z_x} \prod_r \Phi^{\text{join}}(y^r, \mathbf{z}) \prod_i \Phi^{\text{extract}}(z_i, x_i)$$

where the parameter vector θ is used, below, to define the factor Φ^{extract}.

The factors Φ^{join} are deterministic OR operators

$$\Phi^{\text{join}}(y^r, \mathbf{z}) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } y^r = true \wedge \exists i : z_i = r \\ 0 & \text{otherwise} \end{cases}$$

which are included to ensure that the ground fact $r(\mathbf{e})$ is predicted at the aggregate level for the assignment $Y^r = y^r$ only if at least one of the sentence level assignments $Z_i = z_i$ signals a mention of $r(\mathbf{e})$.

The extraction factors Φ^{extract} are given by

$$\Phi^{\text{extract}}(z_i, x_i) \stackrel{\text{def}}{=} \exp\left(\sum_j \theta_j \phi_j(z_i, x_i)\right)$$

where the features ϕ_j are sensitive to the relation name assigned to extraction variable z_i, if any, and cues from the sentence x_i. We will make use of the Mintz *et al.* [112] sentence-level features in the expeiments, as described in Section 2.7.

2.3.3 Discussion

This model was designed to provide a joint approach where extraction decisions are almost entirely driven by sentence-level reasoning. However, defining the Y^r random variables and tying them to the sentence-level variables, Z_i, provides a direct method for modeling weak supervision. We can simply train the model so that the Y variables match the facts in the database, treating the Z_i as hidden variables that can take any value, as long as they produce the correct aggregate predictions.

This approach is related to the multi-instance learning approach of Riedel *et al.* [128], in that both models include sentence-level and aggregate random variables. However, their sentence level variables are binary and they only have a single aggregate variable that takes values $r \in R \cup \{\texttt{none}\}$, thereby ruling out overlapping relations. Additionally, their aggregate decisions make use of Mintz-style aggregate features [112], that collect evidence from multiple sentences, while we use only the deterministic OR nodes. Perhaps surprising, we are still able to improve performance at both the sentential and aggregate extraction tasks.

2.4 Learning

We now present a multi-instance learning algorithm for our weak-supervision model that treats the sentence-level extraction random variables Z_i as latent, and uses facts from a database (*e.g.*, Freebase) as supervision for the aggregate-level variables Y^r.

As input we have (1) Σ, a set of sentences, (2) E, a set of entities mentioned in the sentences, (3) R, a set of relation names, and (4) Δ, a database of atomic facts of the form $r(e_1, e_2)$ for $r \in R$ and $e_i \in E$. Since we are using weak learning, the Y^r variables in $\mathbf{Y}$ are not directly observed, but can be approximated from the database Δ. We use a procedure, **relVector**(e_1, e_2) to return a bit vector whose j^{th} bit is one if $r_j(e_1, e_2) \in \Delta$. The vector does *not* have a bit for the special *none* relation; if there is no relation between the two entities, all bits are zero.

Finally, we can now define the training set to be pairs $\{(\mathbf{x}_i, \mathbf{y}_i) | i = 1 \ldots n\}$, where i is an index corresponding to a particular entity pair (e_j, e_k), $\mathbf{x}_i$ contains all of the sentences with mentions of this pair, and $\mathbf{y}_i =$ **relVector**(e_j, e_k).

Inputs:
(1) Σ, a set of sentences,
(2) E, a set of entities mentioned in the sentences,
(3) R, a set of relation names, and
(4) Δ, a database of atomic facts of the form $r(e_1, e_2)$ for $r \in R$ and $e_i \in E$.

Definitions:
We define the training set $\{(\mathbf{x}_i, \mathbf{y}_i) | i = 1 \dots n\}$, where i is an index corresponding to a particular entity pair (e_j, e_k) in Δ, $\mathbf{x}_i$ contains all of the sentences in Σ with mentions of this pair, and $\mathbf{y}_i =$ **relVector**(e_j, e_k).

Computation:
initialize parameter vector $\mathbf{\Theta} \leftarrow \mathbf{0}$
for $t = 1...T$ **do**
  for $i = 1...n$ **do**
    $(\mathbf{y}', \mathbf{z}') \leftarrow \arg\max_{\mathbf{y},\mathbf{z}} p(\mathbf{y}, \mathbf{z} | \mathbf{x_i}; \theta)$
    if $\mathbf{y}' \neq \mathbf{y_i}$ **then**
      $\mathbf{z}^* \leftarrow \arg\max_{\mathbf{z}} p(\mathbf{z} | \mathbf{x_i}, \mathbf{y_i}; \theta)$
      $\mathbf{\Theta} \leftarrow \mathbf{\Theta} + \phi(\mathbf{x_i}, \mathbf{z}^*) - \phi(\mathbf{x_i}, \mathbf{z}')$
    end if
  end for
end for
Return Θ

Figure 2.2: The MULTIR Learning Algorithm

Given this form of supervision, we would like to find the setting for θ with the highest likelihood:

$$O(\theta) = \prod_i p(\mathbf{y}_i | \mathbf{x}_i; \theta) = \prod_i \sum_{\mathbf{z}} p(\mathbf{y}_i, \mathbf{z} | \mathbf{x}_i; \theta)$$

However, this objective would be difficult to optimize exactly, and algorithms for doing so would be unlikely to scale to data sets of the size we consider. Instead, we make two approximations, described below, leading to a Perceptron-style additive [35] parameter update scheme which has been modified to reason about hidden variables, similar in style to the approaches of [98, 173], but adapted for our specific model. This approximate algorithm is computationally efficient and, as we will see, works well in practice.

Our first modification is to do online learning instead of optimizing the full objective. Define the feature sums $\phi(\mathbf{x}, \mathbf{z}) = \sum_j \phi(x_j, z_j)$ which range over the sentences, as indexed by j. Now, we can define an update based on the gradient of the local log likelihood for example i:

$$\frac{\partial \log O_i(\theta)}{\partial \theta_j} = E_{p(\mathbf{z}|\mathbf{x}_i,\mathbf{y}_i;\theta)}[\phi_j(\mathbf{x}_i,\mathbf{z})] - E_{p(\mathbf{y},\mathbf{z}|\mathbf{x}_i;\theta)}[\phi_j(\mathbf{x}_i,\mathbf{z})]$$

where the deterministic OR Φ^{join} factors ensure that the first expectation assigns positive probability only to assignments that produce the labeled facts $\mathbf{y}_i$ but that the second considers all valid sets of extractions.

Of course, these expectations themselves, especially the second one, would be difficult to compute exactly. Our second modification is to do a Viterbi approximation, by replacing the expectations with maximizations. Specifically, we compute the most likely sentence extractions for the label facts $\arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}_i,\mathbf{y}_i;\theta)$ and the most likely extraction for the input, without regard to the labels, $\arg\max_{\mathbf{y},\mathbf{z}} p(\mathbf{y},\mathbf{z}|\mathbf{x}_i;\theta)$. We then compute the features for these assignments and do a simple additive update. The final algorithm is detailed in Figure 2.2.

2.5 Inference

To support learning, as described above, we need to compute assignments $\arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x},\mathbf{y};\theta)$ and $\arg\max_{\mathbf{y},\mathbf{z}} p(\mathbf{y},\mathbf{z}|\mathbf{x};\theta)$. In this section, we describe algorithms for both cases that use the deterministic OR nodes to simplify the required computations.

Predicting the most likely joint extraction $\arg\max_{\mathbf{y},\mathbf{z}} p(\mathbf{y},\mathbf{z}|\mathbf{x};\theta)$ can be done efficiently given the structure of our model. In particular, the factors Φ^{join} represent deterministic dependencies between $\mathbf{Z}$ and $\mathbf{Y}$, which when satisfied do not affect the probability of the solution. It is thus sufficient to independently compute an assignment for each sentence-level extraction variable Z_i, ignoring the deterministic dependencies. The optimal setting for the aggregate variables $\mathbf{Y}$ is then simply the assignment that is consistent with these extractions. The time complexity is $O(|\mathrm{R}| \cdot |\mathrm{S}|)$.

Predicting sentence level extractions given weak supervision facts, $\arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x},\mathbf{y};\theta)$, is more challenging. We start by computing extraction scores $\Phi^{\text{extract}}(x_i, z_i)$ for each possible extraction assignment $Z_i = z_i$ at each sentence $x_i \in \mathrm{S}$, and storing the values in a dynamic programming table. Next, we must find the most likely assignment $\mathbf{z}$ that respects our output variables $\mathbf{y}$. It turns out that this problem is a variant of the weighted, edge-cover problem, for which there exist

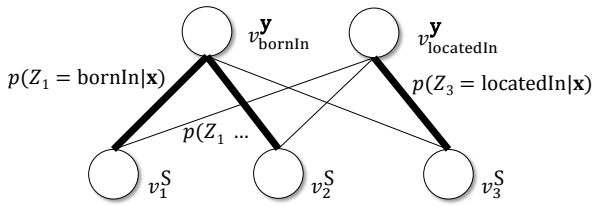


Figure 2.3: Inference of $\arg\max_{\mathbf{z}} p(\mathbf{Z} = \mathbf{z}|\mathbf{x}, \mathbf{y})$ requires solving a weighted, edge-cover problem.

polynomial time optimal solutions.

Let $G = (\mathcal{E}, \mathcal{V} = \mathcal{V}^{\text{S}} \cup \mathcal{V}^{\mathbf{y}})$ be a complete weighted bipartite graph with one node $v_i^{\text{S}} \in \mathcal{V}^{\text{S}}$ for each sentence $x_i \in \text{S}$ and one node $v_r^{\mathbf{y}} \in \mathcal{V}^{\mathbf{y}}$ for each relation $r \in \text{R}$ where $y^r = 1$. The edge weights are given by $c((v_i^{\text{S}}, v_r^{\mathbf{y}})) \stackrel{\text{def}}{=} \Phi^{\text{extract}}(\mathbf{x}_i, z_i)$. Our goal is to select a subset of the edges which maximizes the sum of their weights, subject to each node $v_i^{\text{S}} \in \mathcal{V}^{\text{S}}$ being incident to exactly one edge, and each node $v_r^{\mathbf{y}} \in \mathcal{V}^{\mathbf{y}}$ being incident to at least one edge.

Exact Solution An exact solution can be obtained by first computing the maximum weighted bipartite matching, and adding edges to nodes which are not incident to an edge. This can be computed in time $O(|\mathcal{V}|(|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|))$, which we can rewrite as $O((|\text{R}| + |\text{S}|)(|\text{R}||\text{S}| + (|\text{R}| + |\text{S}|) \log(|\text{R}| + |\text{S}|)))$.

Approximate Solution An approximate solution can be obtained by iterating over the nodes in $\mathcal{V}^{\mathbf{y}}$, and each time adding the highest weight incident edge whose addition doesn't violate a constraint. The running time is $O(|\text{R}||\text{S}|)$. This greedy search guarantees each fact is extracted at least once and allows any additional extractions that increase the overall probability of the assignment. Given the computational advantage, we use it in all of the experimental evaluations.

2.6 Experimental Setup

We follow the approach of Riedel *et al.* [128] for generating weak supervision data, computing features, and evaluating aggregate extraction. We also introduce new metrics for measuring sentential extraction performance, both relation-independent and relation-specific.

2.6.1 Data Generation

We used the same data sets as Riedel *et al.* [128] for weak supervision. The data was first tagged with the Stanford NER system [60] and then entity mentions were found by collecting each continuous phrase where words were tagged identically (*i.e.*, as a person, location, or organization). Finally, these phrases were matched to the names of Freebase entities.

Given the set of matches, define Σ to be set of NY Times sentences with two matched phrases, E to be the set of Freebase entities which were mentioned in one or more sentences, Δ to be the set of Freebase facts whose arguments, e_1 and e_2 were mentioned in a sentence in Σ, and R to be set of relations names used in the facts of Δ. These sets define the weak supervision data.

2.6.2 Features and Initialization

We use the set of sentence-level features described by Riedel *et al.* [128], which were originally developed by Mintz *et al.* [112]. These include indicators for various lexical, part of speech, named entity, and dependency tree path properties of entity mentions in specific sentences, as computed with the Malt dependency parser [114] and OpenNLP POS tagger[1]. However, unlike the previous work, we did not make use of any features that explicitly aggregate these properties across multiple mention instances.

The MULTIR algorithm has a single parameter T, the number of training iterations, that must be specified manually. We used $T = 50$ iterations, which performed best in development experiments.

2.6.3 Evaluation Metrics

Evaluation is challenging, since only a small percentage (approximately 3%) of sentences match facts in Freebase, and the number of matches is highly unbalanced across relations, as we will see in more detail later. We use the following metrics.

Aggregate Extraction Let Δ^e be the set of extracted relations for any of the systems; we compute aggregate precision and recall by comparing Δ^e with Δ. This metric is easily computed but

[1] http://opennlp.sourceforge.net/

underestimates extraction accuracy because Freebase is incomplete and some true relations in Δ^e will be marked wrong.

Sentential Extraction Let S^e be the sentences where some system extracted a relation and S^F be the sentences that match the arguments of a fact in Δ. We manually compute sentential extraction accuracy by sampling a set of 1000 sentences from $S^e \cup S^F$ and manually labeling the correct extraction decision, either a relation $r \in R$ or $none$. We then report precision and recall for each system on this set of sampled sentences. These results provide a good approximation to the true precision but can overestimate the actual recall, since we did not manually check the much larger set of sentences where no approach predicted extractions.

2.6.4 Precision / Recall Curves

To compute precision / recall curves for the tasks, we ranked the MULTIR extractions as follows. For sentence-level evaluations, we ordered according to the extraction factor score $\Phi^{\text{extract}}(z_i, x_i)$. For aggregate comparisons, we set the score for an extraction $Y^r = true$ to be the max of the extraction factor scores for the sentences where r was extracted.

2.7 Experiments

To evaluate our algorithm, we first compare it to an existing approach for using multi-instance learning with weak supervision [128], using the same data and features. We report both aggregate extraction and sentential extraction results. We then investigate relation-specific performance of our system. Finally, we report running time comparisons.

2.7.1 Aggregate Extraction

Figure 2.4 shows approximate precision / recall curves for three systems computed with aggregate metrics (Section 2.6.3) that test how closely the extractions match the facts in Freebase. The systems include the original results reported by Riedel *et al.* [128] as well as our new model (MULTIR). We also compare with SOLOR, a reimplementation of their algorithm, which we built in Factorie [105], and will use later to evaluate sentential extraction.

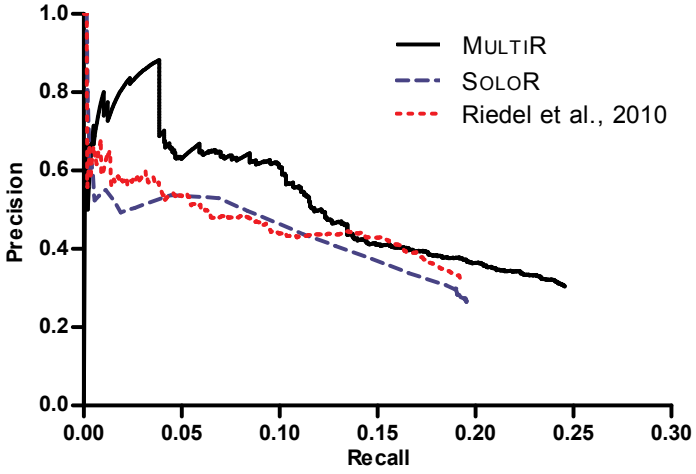


Figure 2.4: Aggregate extraction precision / recall curves for Riedel *et al.* [128], a reimplementation of that approach (SOLOR), and our algorithm (MULTIR).

MULTIR achieves competitive or higher precision over all ranges of recall, with the exception of the very low recall range of approximately 0-1%. It also significantly extends the highest recall achieved, from 20% to 25%, with little loss in precision. To investigate the low precision in the 0-1% recall range, we manually checked the ten highest confidence extractions produced by MULTIR that were marked wrong. We found that all ten were true facts that were simply missing from Freebase. A manual evaluation, as we perform next for sentential extraction, would remove this dip.

2.7.2 Sentential Extraction

Although their model includes variables to model sentential extraction, Riedel *et al.* [128] did not report sentence level performance. To generate the precision / recall curve we used the joint model assignment score for each of the sentences that contributed to the aggregate extraction decision.

Figure 2.4 shows approximate precision / recall curves for MULTIR and SOLOR computed against manually generated sentence labels, as defined in Section 2.6.3. MULTIR achieves significantly higher recall with a consistently high level of precision. At the highest recall point, MULTIR reaches 72.4% precision and 51.9% recall, for an F1 score of 60.5%.

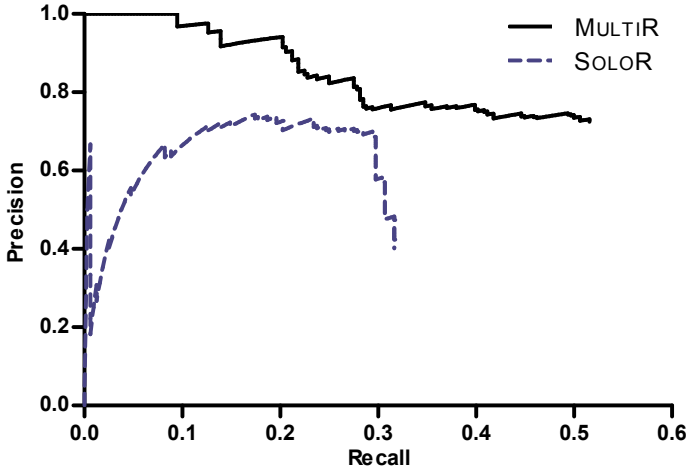


Figure 2.5: Sentential extraction precision / recall curves for MULTIR and SOLOR.

2.7.3 Relation-Specific Performance

Since the data contains an unbalanced number of instances of each relation, we also report precision and recall for each of the ten most frequent relations. Let S_r^M be the sentences where MULTIR extracted an instance of relation $r \in R$, and let S_r^F be the sentences that match the arguments of a fact about relation r in Δ. For each r, we sample 100 sentences from both S_r^M and S_r^F and manually check accuracy. To estimate precision $\tilde{P}_r$ we compute the ratio of true relation mentions in S_r^M, and to estimate recall $\tilde{R}_r$ we take the ratio of true relation mentions in S_r^F which are returned by our system.

Table 2.1 presents this approximate precision and recall for MULTIR on each of the relations, along with statistics we computed to measure the quality of the weak supervision. Precision is high for the majority of relations but recall is consistently lower. We also see that the Freebase matches are highly skewed in quantity and can be low quality for some relations, with very few of them actually corresponding to true extractions. The approach generally performs best on the relations with a sufficiently large number of true matches, in many cases even achieving precision that outperforms the accuracy of the heuristic matches, at reasonable recall levels.

Relation	Freebase Matches		MULTIR	
	#sents	% true	$\tilde{P}$	$\tilde{R}$
/business/person/company	302	89.0	100.0	25.8
/people/person/place_lived	450	60.0	80.0	6.7
/location/location/contains	2793	51.0	100.0	56.0
/business/company/founders	95	48.4	71.4	10.9
/people/person/nationality	723	41.0	85.7	15.0
/location/neighborhood/neighborhood_of	68	39.7	100.0	11.1
/people/person/children	30	80.0	100.0	8.3
/people/deceased_person/place_of_death	68	22.1	100.0	20.0
/people/person/place_of_birth	162	12.0	100.0	33.0
/location/country/administrative_divisions	424	0.2	N/A	0.0

Table 2.1: Estimated precision and recall by relation, as well as the number of matched sentences (#sents) and accuracy (% true) of matches between sentences and facts in Freebase.

2.7.4 Overlapping Relations

Table 2.1 also highlights some of the effects of learning with overlapping relations. For example, in the data, almost all of the matches for the administrative_divisions relation overlap with the contains relation, because they both model relationships for a pair of locations. Since, in general, sentences are much more likely to describe a contains relation, this overlap leads to a situation were almost none of the administrate_division matches are true ones, and we cannot accurately learn an extractor. However, we can still learn to accurately extract the contains relation, despite the distracting matches. Similarly, the place_of_birth and place_of_death relations tend to overlap, since it is often the case that people are born and die in the same city. In both cases, the precision outperforms the labeling accuracy and the recall is relatively high.

To measure the impact of modeling overlapping relations, we also evaluated a simple, restricted baseline. Instead of labeling each entity pair with the set of all true Freebase facts, we created a dataset where each true relation was used to create a different training example. Training MULTIR on this data simulates effects of conflicting supervision that can come from not modeling overlaps. On average across relations, precision increases 12 points but recall drops 26 points, for an overall reduction in F1 score from 60.5% to 40.3%.

2.7.5 Running Time

One final advantage of our model is the modest running time. Our implementation of the Riedel *et al.* [128] approach required approximately 6 hours to train on NY Times 05-06 and 4 hours to test on the NY Times 07, each without preprocessing. Although they do sampling for inference, the global aggregation variables require reasoning about an exponentially large (in the number of sentences) sample space.

In contrast, our approach required approximately one minute to train and less than one second to test, on the same data. This advantage comes from the decomposition that is possible with the deterministic OR aggregation variables. For test, we simply consider each sentence in isolation and during training our approximation to the weighted assignment problem is linear in the number of sentences.

2.7.6 Discussion

The sentential extraction results demonstrates the advantages of learning a model that is primarily driven by sentence-level features. Although previous approaches have used more sophisticated features for aggregating the evidence from individual sentences, we demonstrate that aggregating strong sentence-level evidence with a simple deterministic OR that models overlapping relations is more effective, and also enables training of a sentence extractor that runs with no aggregate information.

While the Riedel *et al.* approach does include a model of which sentences express relations, it makes significant use of aggregate features that are primarily designed to do entity-level relation predictions and has a less detailed model of extractions at the individual sentence level. Perhaps surprisingly, our model is able to do better at both the sentential and aggregate levels.

2.8 Related Work

Supervised-learning approaches to IE were introduced in [147]. While they offer high precision and recall, these methods are unlikely to scale to the thousands of relations found in text on the Web. Open IE systems, which perform self-supervised learning of relation-independent extractors (*e.g.*, Preemptive IE [141], TEXTRUNNER [11, 12] and WOE [170]) can scale to millions of documents,

but don't output canonicalized relations.

2.8.1 Weak Supervision

Weak supervision (also known as distant- or self supervision) refers to a broad class of methods, but we focus on the increasingly-popular idea of using a store of structured data to heuristicaly label a textual corpus. Craven and Kumlien [38] introduced the idea by matching the Yeast Protein Database (YPD) to the abstracts of papers in PubMed and training a naive-Bayes extractor. Bellare and McCallum [14] used a database of BibTex records to train a CRF extractor on 12 bibliographic relations. The KYLIN system aplied weak supervision to learn relations from Wikipedia, treating infoboxes as the associated database [168]; Wu *et al.* [169] extended the system to use smoothing over an automatically generated infobox taxonomy. Mintz *et al.* [112] used Freebase facts to train 100 relational extractors on Wikipedia. Hoffmann *et al.* [81] describe a system similar to KYLIN, but which dynamically generates lexicons in order to handle sparse data, learning over 5000 Infobox relations with an average F1 score of 61%. Yao *et al.* [171] perform weak supervision, while using selectional preference constraints to a jointly reason about entity types.

The NELL system [24] can also be viewed as performing weak supervision. Its initial knowledge consists of a selectional preference constraint and 20 ground fact seeds. NELL then matches entity pairs from the seeds to a Web corpus, but instead of learning a probabilistic model, it bootstraps a set of extraction patterns using semi-supervised methods for multitask learning.

2.8.2 Multi-Instance Learning

Multi-instance learning was introduced in order to combat the problem of ambiguously-labeled training data when predicting the activity of different drugs [46]. Bunescu and Mooney [20] connect weak supervision with multi-instance learning and extend their relational extraction kernel to this context.

Riedel *et al.* [128], combine weak supervision and multi-instance learning in a more sophisticated manner, training a graphical model, which assumes only that *at least one* of the matches between the arguments of a Freebase fact and sentences in the corpus is a true relational mention. Our model may be seen as an extension of theirs, since both models include sentence-level and

aggregate random variables. However, Riedel *et al.* have only a single aggregate variable that takes values $r \in R \cup \{\texttt{none}\}$, thereby ruling out overlapping relations. We have discussed the comparison in more detail throughout the chapter, including in the model formulation section and experiments.

2.9 Conclusion

Weak supervision enables us to dramatically reduce the human effort necessary to create relation extractors. Often, that can be reduced to just selecting a database of relation instances, but the approach is not always applicable. Since the process of matching database tuples to sentences is inherently heuristic, researchers have proposed multi-instance learning algorithms as a means for coping with the resulting noisy data. Unfortunately, previous approaches assume that all relations are *disjoint* — for example they cannot extract the pair `isCEOof(Cook, Apple)` and `isShareholderOf(Jobs, Apple)`, because two relations are not allowed to have the same arguments.

This chapter presented a novel approach for multi-instance learning with overlapping relations that combines a sentence-level extraction model with a simple, corpus-level component for aggregating the individual facts. We applied our model to learn extractors for NY Times text using weak supervision from Freebase. Experiments showed improvements for both sentential and aggregate (corpus level) extraction, and demonstrated that the approach is computationally efficient.

Chapter 3

LUCHS: WEAKLY SUPERVISED EXTRACTION OF SPARSE RELATIONS USING WEB-SCALE LEXICON INDUCTION

Learning with weak supervision promises to dramatically reduce the human effort necessary to create relation extractors. Often, the effort can be reduced to no more than providing a database of relation instances. Unfortunately, that does not always work. As we have seen in the previous chapter, one problem is the uncertainty in the heuristic labeling. For many relations, however, we are also facing a problem of sparsity when only few sentences can be heuristically labeled.

This chapter presents a novel approach for weakly supervised relation extraction which learns 5025 relational extractors — more than an order of magnitude greater than any previous approach — with an average F1 score of 61%. Crucial to LUCHS's performance is an automated system for *dynamic lexicon learning*, which allows it to learn accurately from heuristically-generated training data, which is often noisy and sparse. LUCHS uses a large Web crawl to induce sets of relation-specific lexicons, and uses Wikipedia infoboxes as a weak signal for supervision.

3.1 Introduction

Knowledge-based weak supervision has been used to learn relational extractors in a variety of domains. Several systems [112, 127, 81] extract from news articles using Freebase for supervision. Others extract from publication abstracts using biological databases for supervision [38].

With the Kylin system Wu and Weld [168] applied this idea to Wikipedia. A sizable fraction of Wikipedia articles have associated infoboxes — relational summaries of the key aspects of the subject of the article. For example, the infobox for Alan Turing's Wikipedia page lists the values of 10 attributes, including his birthdate, nationality and doctoral advisor. The Kylin system matches values of an article's infobox attributes to corresponding sentences in the article, and [168] suggested that their approach could extract thousands of relations [167]. Unfortunately, however, they never tested the idea on more than a dozen relations. Indeed, no one has demonstrated a practical way to

extract more than about one hundred relations.

We note that Wikipedia's infobox 'ontology' is a particularly interesting target for extraction. As a by-product of thousands of contributors, it is broad in coverage and growing quickly. Unfortunately, the schemata are surprisingly noisy and most are sparsely populated; challenging conditions for extraction.

This chapter presents LUCHS, an autonomous, weakly supervised system, which learns 5025 relational extractors — an order of magnitude greater than any previous effort. Like Kylin, LUCHS creates training data by matching Wikipedia attribute values with corresponding sentences, but by itself, this method was insufficient for accurate extraction of most relations. Thus, LUCHS introduces a new technique, *dynamic lexicon features*, which dramatically improves performance when learning from sparse data and that way enables scalability.

Figure 3.1 summarizes the architecture of LUCHS. At the highest level, LUCHS's offline training process resembles that of Kylin. Wikipedia pages containing infoboxes are used to train a classifier that can predict the appropriate schema for pages missing infoboxes. Additionally, the values of infobox attributes are compared with article sentences to heuristically generate training data. LUCHS's major innovation is a feature-generation process, which starts by harvesting HTML lists from a 5B document Web crawl, discarding 98% to create a set of 49M semantically-relevant lists. When learning an extractor for relation R, LUCHS extracts seed phrases from R's training data and uses a semi-supervised learning algorithm to create several relation-specific lexicons at different points on a precision-recall spectrum. These lexicons form Boolean features which, along with lexical and dependency parser-based features, are used to produce a CRF extractor for each relation — one which performs much better than lexicon-free extraction on sparse training data.

At runtime, LUCHS feeds pages to the article classfier, which predicts which infobox schema is most appropriate for extraction. Then a small set of relation-specific extractors are applied to each sentence, outputting tuples. Our experiments demonstrate a high F1 score, 61%, across the 5025 relational extractors learned.

This chapter makes several contributions:

- We present LUCHS, a weakly supervised IE system capable of learning more than an order of magnitude more relation-specific extractors than previous systems.

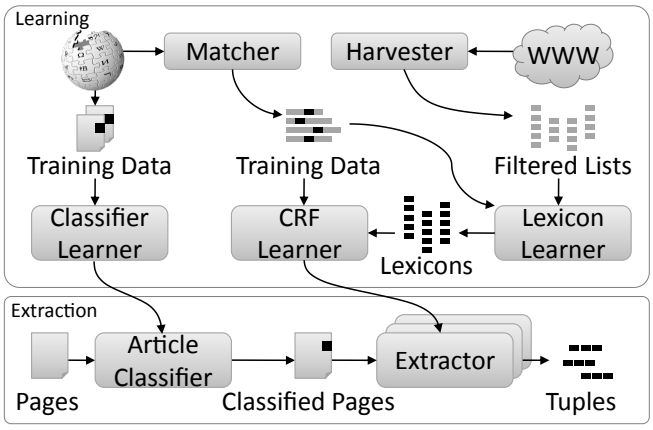


Figure 3.1: Architecture of LUCHS. In order to handle sparsity in its heuristically-generated training data, LUCHS generates custom lexicon features when learning each relational extractor.

- We describe the construction and use of *dynamic lexicon features*, a novel technique, that enables hyper-lexicalized extractors which cope effectively with sparse training data.
- We evaluate the overall end-to-end performance of LUCHS, showing an F1 score of 61% when extracting relations from randomly selected Wikipedia pages.
- We present a comprehensive set of additional experiments, evaluating LUCHS's individual components, measuring the effect of dynamic lexicon features, testing sensitivity to varying amounts of training data, and categorizing the types of relations LUCHS can extract.

3.2 Weak Supervision from Wikipedia Infoboxes

Wikipedia is an ideal starting point for our long-term goal of creating a massive knowledge base of extracted facts for two reasons. First, it is comprehensive, containing a diverse body of content with significant depth. Perhaps more importantly, Wikipedia's structure facilitates weakly supervised extraction. Infoboxes are short, manually-created tabular summaries of many articles' key facts — effectively defining a relational schema for that class of entity. Since the same facts are often expressed in both article and ontology, matching values of the ontology to the article can deliver valuable, though noisy, training data.

For example, the Wikipedia article on "Jerry Seinfeld" contains the sentence "Seinfeld was born in Brooklyn, New York." and the article's infobox contains the attribute "birth_place = Brooklyn".

By matching the attribute's value "Brooklyn" to the sentence, we heuristically generate training data for a birth_place extractor.

Note that this approach differs from MULTIR's. Here, we only match a single value and assume that the first argument of the relation is the main entity of the article. We also do not constrain matches to the output of a named entity tagger.

3.3 Learning Extractors

We first assume that each Wikipedia infobox attribute corresponds to a unique relation (but see Section 3.5.6) for which we would like to learn a specific extractor. A major challenge with such an approach is scalability. Running a relation-specific extractor for each of Wikipedia's 34,000 unique infobox attributes on each of Wikipedia's 50 million sentences would require 1.7 trillion extractor executions.

We therefore choose a *hierarchical* approach that combines both article classifiers and relation extractors. For each infobox schema, LUCHS trains a classifier that predicts if an article is likely to contain that schema. Only when an article is likely to contain a schema, does LUCHS run that schema's relation extractors. To extract infobox attributes from all of Wikipedia, LUCHS now needs orders of magnitude fewer executions.

While this approach does not propagate information from extractors back to article classifiers, experiments confirm that our article classifiers nonetheless deliver accurate results (Section 3.5.2), reducing the potential benefit of joint inference. In addition, our approach reduces the need for extractors to keep track of the larger context, thus simplifying the extraction problem.

We briefly summarize article classification: We use a linear, multi-class classifier with six kinds of features: words in the article title, words in the first sentence, words in the first sentence which are direct objects to the verb 'to be', article section headers, Wikipedia categories, and their ancestor categories. We use the voted perceptron algorithm [64] for training.

More challenging are the attribute extractors, which we wish to be simple, fast, and able to well capture local dependencies. We use a linear-chain conditional random field (CRF) — an undirected graphical model connecting a sequence of input and output random variables, $x = (x_0, \ldots, x_T)$ and $y = (y_0, \ldots, y_T)$ [92]. Input variables are assigned words w. The states of output variables

represent discrete labels l, e.g. Arg$_i$-of-Rel$_j$ and Other. In our case, variables are connected in a chain, following the first-order Markov assumption. We train to maximize conditional likelihood of output variables given an input probability distribution. The CRF models $p(y|x)$ are represented with a log-linear distribution

$$p(y|x) = \frac{1}{Z(x)} \exp \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, x, t)$$

where feature functions, f, encode sufficient statistics of (x, y), T is the length of the sequence, K is the number of feature functions, and λ_k are parameters representing feature weights, which we learn during training. $Z(x)$ is a partition function used to normalize the probabilities to 1. Feature functions allow complex, overlapping global features with lookahead.

Common techniques for learning the weights λ_k include numeric optimization algorithms such as stochastic gradient descent or L-BFGS. In our experiments, we again use the simpler and more efficient voted-perceptron algorithm [35]. The linear-chain layout enables efficient interence using the dynamic programming-based Viterbi algorithm [92].

We evaluate nine kinds of Boolean features:

Words For each input word w we introduce feature $f_w^{\mathbf{w}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[x_t=w]}$.

State Transitions For each transition between output labels l_i, l_j we add feature $f_{l_i,l_j}^{\mathbf{tran}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[y_{t-1}=l_i \wedge y_t=l_j]}$.

Word Contextualization For parameters p and s we add features $f_w^{\mathbf{prev}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[w \in \{x_{t-p},\ldots,x_{t-1}\}]}$ and $f_w^{\mathbf{sub}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[w \in \{x_{t+1},\ldots,x_{t+s}\}]}$ which capture a window of words appearing before and after each position t.

Capitalization We add feature $f^{\mathbf{cap}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[x_t \text{is capitalized}]}$.

Digits We add feature $f^{\mathbf{dig}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[x_t \text{is digits}]}$.

Dependencies We set $f^{\mathbf{dep}}(y_{t-1}, y_t, x, t)$ to the lemmatized sequence of words from x_t to the root of the dependency tree, computed using the Stanford parser [103].

First Sentence We set $f^{\mathbf{fs}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[x_t \text{in first sentence of article}]}$.

Gaussians For numeric attributes, we fit a Gaussian (μ, σ) and add feature $f_i^{\mathbf{gau}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[|x_t - \mu| < i\sigma]}$ for parameters i.

Lexicons For non-numeric attributes, and for a lexicon l, *i.e.* a set of related words, we add feature $f_l^{\mathbf{lex}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[x_t \in l]}$. Lexicons are explained in the following section.

3.4 Extraction with Lexicons

It is often possible to group words that are likely to be assigned similar labels, even if many of these words do not appear in our training set. The obtained lexicons then provide an elegant way to improve the generalization ability of an extractor, especially when only little training data is available. However, there is a danger of overfitting, which we discuss in Section 3.4.2.

The next section explains how we mine the Web to obtain a large corpus of quality lists. Then Section 3.4.2 presents our semi-supervised algorithm for learning semantic lexicons from these lists.

3.4.1 Harvesting Lists from the Web

Domain-independence requires access to an extremely large number of lists, but our tight integration of lexicon acquisition and CRF learning requires that relevant lists be accessed instantaneously. Approaches using search engines or wrappers at query time [56, 164] are too slow; we must extract and index lists prior to learning.

We begin with a 5 billion page Web crawl. LUCHS can be combined with any list harvesting technique, but we choose a simple approach, extracting lists defined by HTML `<ul>` or `<ol>` tags. The set of lists obtained in this way is extremely noisy — many lists comprise navigation bars, tag sets, spam links, or a series of long text paragraphs. This is consistent with the observation that less than 2% of Web tables are relational [22].

We therefore apply a series of filtering steps. We remove lists of only one or two items, lists containing long phrases, and duplicate lists from the same host. After filtering we obtain 49 million lists, containing 56 million unique phrases.

3.4.2 Semi-Supervised Learning of Lexicons

While training a CRF extractor for a given relation, LUCHS uses its corpus of lists to automatically generate a set of semantic lexicons — *specific to that relation*. The technique proceeds in three steps, which have been engineered to run extremely quickly:

1. Seed phrases are extracted from the labeled training set.

2. A learning algorithm expands the seed phrases into a set of lexicons.

3. The semantic lexicons are added as features to the CRF learning algorithm.

Extracting Seed Phrases

For each training sentence LUCHS first identifies subsequences of labeled words, and for each such labeled subsequence, LUCHS creates one or more seed phrases p. Typically, a set of seeds consists precisely of the labeled subsequences. However, if the labeled subsequences are long and have substructure, *e.g.*, 'San Remo, Italy', our system splits at the separator token, and creates additional seed sets from prefixes and postfixes.

From Seeds to Lexicons

To expand a set of seeds into a lexicon, LUCHS must identify relevant lists in the corpus. Relevancy can be computed by defining a similarity between lists using the vector-space model. Specifically, let $\mathcal{L}$ denote the corpus of lists, and $\mathcal{P}$ be the set of unique phrases from $\mathcal{L}$. Each list $l^0 \in \mathcal{L}$ can be represented as a vector of weighted phrases $p \in \mathcal{P}$ appearing on the list, $l^0 = (l^0_{p_1} l^0_{p_2} \ldots l^0_{p_{|\mathcal{P}|}})$. Following the notion of *inverse document frequency,* a phrase's weight is inversely proportional to the number of lists containing the phrase. Popular phrases which appear on many lists thus receive

a small weight, whereas rare phrases are weighted higher:

$$l^0_{p_i} = \frac{1}{|\{l \in \mathcal{L} | p \in l\}|}$$

Unlike the vector space model for documents, we ignore term frequency, since the vast majority of lists in our corpus don't contain duplicates. This vector representation supports the simple cosine definition of *list similarity*, which for lists $l^0, l^1 \in \mathcal{L}$ is defined as

$$sim_{cos} := \frac{l^0 \cdot l^1}{\|l^0\| \|l^1\|}.$$

Intuitively, two lists are similar if they have many overlapping phrases, the phrases are not too common, and the lists don't contain many other phrases. By representing the seed set as another vector, we can find similar lists, hopefully containing related phrases. We then create a semantic lexicon by collecting phrases from a range of related lists.

For example, one lexicon may be created as the union of all phrases on lists that have non-zero similarity to the seed list. Unfortunately, due to the noisy nature of the Web lists such a lexicon may be very large and may contain many irrelevant phrases. We expect that lists with higher similarity are more likely to contain phrases which are related to our seeds; hence, by varying the similarity threshold one may produce lexicons representing different compromises between lexicon precision and recall. Not knowing which lexicon will be most useful to the extractors, LUCHS generates several and lets the extractors learn appropriate weights.

However, since list similarities vary depending on the seeds, fixed thresholds are not an option. If *#similarlists* denotes the number of lists that have non-zero similarity to the seed list and *#lexicons* the total number of lexicons we want to generate, LUCHS sets lexicon $i \in \{0, \ldots, \#lexicons - 1\}$ to be the union of prases on the

$$\textit{\#similarlists}^{i/\textit{\#lexicons}}$$

most similar lists.[1]

[1]For practical reasons, we exclude the case $i =$ #lexicons in our experiments.

Efficiently Creating Lexicons

We create lexicons from lists that are *similar* to our seed vector, so we only consider lists that have at least one phrase in common. Importantly, our index structures allow LUCHS to select the relevant lists efficiently. For each seed, LUCHS retrieves the set of containing lists as a sorted sequence of list identifiers. These sequences are then merged yielding a sequence of list identifiers with associated seed-hit counts. Precomputed list lengths and inverse document frequencies are also retrieved from indices, allowing efficient computation of similarity. The worst case complexity is $O(\log(S)SK)$ where S is the number of seeds and K the maximum number of lists to consider per seed.

Preventing Lexicon Overfitting

Finally, we integrate the acquired semantic lexicons as features into the CRF. Although Section 3.3 discussed how to use lexicons as CRF features, there are some subtleties. Recall that the lexicons were created from seeds extracted from the training set. If we now train the CRF on the *same* examples that generated the lexicon features, then the CRF will likely overfit, and weight the lexicon features too highly!

Before training, we therefore split the training set into k partitions. For each example in a partition we assign features based on lexicons generated from only the $k-1$ *remaining partitions*. This avoids overfitting and ensures that we will not perform much worse than without lexicon features. When we apply the CRF to our test set, we use the lexicons based on all k partitions. We refer to this technique as *cross-training*.

3.5 Experiments

We start by evaluating end-to-end performance of LUCHS when applied to Wikipedia text, then analyze the characteristics of its components. Our experiments use the 10/2008 English Wikipedia dump.

3.5.1 Overall Extraction Performance

To evaluate the end-to-end performance of LUCHS, we test the pipeline which first classifies incoming pages, activating a small set of extractors on the text. To ensure adequate training and test data,

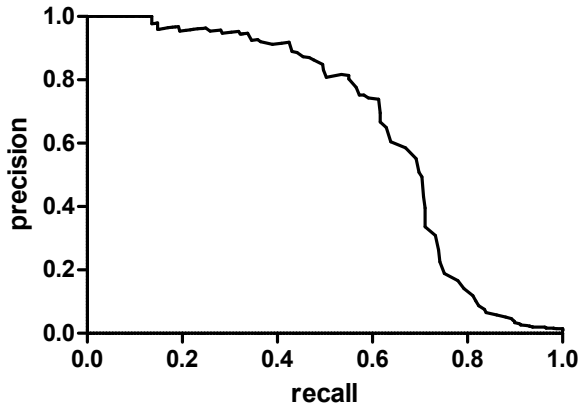


Figure 3.2: Precision / recall curve for end-to-end system performance on 100 random articles.

we limit ourselves to infobox classes with at least ten instances; there exist 1,583 such classes, together comprising 981,387 articles. We only consider the first ten sentences for each article, and we only consider 5025 attributes.[2] We create a test set by sampling 100 articles randomly; these articles are not used to train article classifiers or extractors. Each test article is then automatically classified, and a random attribute of the predicted schema is selected for extraction. Gold labels for the selected attribute and article are created manually by a human judge and compared to the token-level predictions from the extractors which are trainined on the remaining articles with heuristic matches.

Overall, LUCHS reaches a precision of .55 at a recall of .68, giving an F1-score of .61 (Figure 3.2). Analyzing the errors in more detail, we find that in 11 of 100 cases an article was incorrectly classified. We note that in at least two of these cases the predicted class could also be considered correct. For example, instead of *Infobox Minor Planet* the extractor predicted *Infobox Planet*.

On five of the selected attributes the extractor failed because the attributes could be considered unlearnable: The flexibility of Wikipedia's infobox system allows contributors to introduce attributes for formatting, for example defining element order. In the future we wish to train LUCHS to ignore this type of attribute.

We also compared the heuristic matches contained in the selected 100 articles to the gold stan-

[2]Attributes were selected to have at least 10 heuristic matches, to have 10% of values covered by matches, and 10% of articles with attribute in infobox covered by matches.

dard: The matches reach a precision of .90 at a recall of .33, giving an F1-score of .48. So while most heuristic matches hit mentions of attribute values, many other mentions go unmatched. Manual analysis shows that these values are often missing from an infobox, are formatted differently, or are inconsistent to what is stated in the article.

So why did the low recall of the heuristic matches not adversely affect recall of our extractors? For most articles, an attribute can be assigned a single unique value. When training an attribute extractor, only articles that contained a heuristic match for that attribute were considered, thus avoiding many cases of unmatched mentions.

Subsequent experiments evaluate the performance of LUCHS components in more detail.

3.5.2 Article Classification

The first step in LUCHS's run-time pipeline is determining which infobox schemata are most likely to be found in a given article. To test this, we randomly split our 981,387 articles into 4/5 for training and 1/5 for testing, and train a single multi-class classifier. For this experiment, we use the original infobox class of an article as its gold label. We compute the accuracy of the prediction at .92. Since some classes can be considered interchangeable, this number represents a lower bound on performance.

3.5.3 Factors Affecting Extraction Accuracy

We now evaluate attribute extraction assuming perfect article classification. To keep training time manageable, we sample 100 articles for training and 100 articles for testing[3] for each of 100 random attributes. We again only consider the first ten sentences of each article, and we only consider articles that have heuristic matches with the attribute. We measure F1-score at a token-level, taking the heuristic matches as ground-truth.

We first test the performance of extractors trained using our basic features (Section 3.3)[4], not including lexicons and Gaussians. We begin using word features and obtain a token-level F1-score of .311 for text and .311 for numeric attributes. Adding any of our additional features improves these

[3]These numbers are smaller for attributes with less training data available, but the same split is maintained.

[4]For contextualization features we choose $p, s = 5$.

Features	F1-Score
Text attributes	
Baseline	.491
Baseline + Lexicons w/o CT	.367
Baseline + Lexicons	.545
Numeric attributes	
Baseline	.586
Baseline + Gaussians w/o CT	.623
Baseline + Gaussians	.627

Table 3.1: Impact of Lexicon and Gaussian features. Cross-Training (CT) is essential to improve performance.

scores, but the relative improvements vary: For both text and numeric attributes, contextualization and dependency features deliver the largest improvement. We then iteratively add the feature with largest improvement until no further improvement is observed. We finally obtain an F1-score of .491 for text and .586 for numeric attributes. For text attributes the extractor uses word, contextualization, first sentence, capitalization, and digit features; for numeric attributes the extractor uses word, contextualization, digit, first sentence, and dependency features. We use these extractors as a baseline to evaluate our lexicon and Gaussian features.

Varying the size of the training sets affects results: Taking more articles raises the F1-score, but taking more sentences per article reduces it. This is because Wikipedia articles often summarize a topic in the first few paragraphs and later discuss related topics, necessitating reference resolution which we plan to add in future work.

3.5.4 Lexicon and Gaussian Features

We next study how our distribution features[5] impact the quality of the baseline extractors (Table 3.1). Without cross-training we observe a reduction in performance, due to overfitting. Cross-training avoids this, and substantially improves results over the baseline. While cross-training is particularly critical for lexicon features, it is less needed for Gaussians where only two parameters, mean and deviation, are fitted to the training set.

The relative improvements depend on the number of available training examples (Table 3.2).

[5]We set the number of lexicon and Gaussian features to 4.

# Train	F1-B	F1-LUCHS	ΔF1	ΔPr	ΔRe
Text attributes					
10	.379	.439	+16%	+10%	+20%
25	.447	.504	+13%	+7%	+20%
100	.491	.545	+11%	+5%	+17%
Numeric attributes					
10	.484	.531	+10%	+4%	+13%
25	.552	.596	+8%	+4%	+10%
100	.586	.627	+7%	+5%	+8%

Table 3.2: Lexicon and Gaussian features greatly expand F1 score (F1-LUCHS) over the baseline (F1-B), in particular for attributes with few training examples. Gains are mainly due to increased recall.

Lexicon and Gaussian features especially benefit extractors for sparse attributes. Here we can also see that the improvements are mainly due to increases in recall.

3.5.5 Scaling to All of Wikipedia

Finally, we take our best extractors and run them on all 5025 attributes, again assuming perfect article classification and using heuristic matches as gold-standard. Figure 3.3 shows the distribution of obtained F1 scores. 810 text attributes and 328 numeric attributes reach a score of 0.80 or higher.

The performance depends on the number of available training examples, and that number is governed by a long-tailed distribution. For example, 61% of the attributes in our set have 50 or fewer examples, 36% have 20 or fewer. Interestingly, the number of training examples had a smaller effect on performance than expected. Figure 3.4 shows the correlation between these variables. Lexicon and Gaussian features enables acceptable performance even for sparse attributes.

Averaging across all attributes we obtain F1 scores of 0.56 and 0.60 for textual and numeric values respectively. We note that these scores assume that all attributes are equally important, weighting rare attributes just like common ones. If we weight scores by the number of attribute instances, we obtain F1 scores of 0.64 (textual) and 0.78 (numeric). In each case, precision is slightly higher than recall.

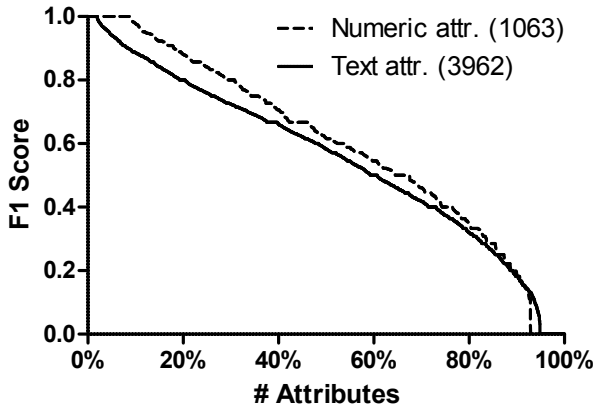


Figure 3.3: F1 scores among attributes, ranked by score. 810 text attributes (20%) and 328 numeric attributes (31%) had an F1-score of .80 or higher.

3.5.6 Towards an Attribute Ontology

The true promise of relation-specific extractors comes when an ontology ties the system together. By learning a probabilistic model of selectional preferences, one can use joint inference to improve extraction accuracy. One can also answer scientific questions, such as "How many of the learned Wikipedia attributes are distinct?" It is clear that many duplicates exist due to collaborative sloppiness, but semantic similarity is a matter of opinion and an exact answer is impossible.

Nevertheless, we clustered the textual attributes in several ways. First, we cleaned the attribute names heuristically and performed spell check. The "distance" between two attributes was calculated with a combination of edit distance and IR metrics with Wordnet synonyms; then hierarchical agglomerative clustering was performed. We manually assigned names to the clusters and cleaned them, splitting and joining as needed. The result is too crude to be called an ontology, but we continue its elaboration. There are a total of 3962 attributes grouped in about 1282 clusters (not yet counting attributes with numerical values); the largest cluster, *location*, has 115 similar attributes. Figure 3.5 shows the confusion matrix between attributes in the biggest clusters; the shade of the i, j^{th} pixel indicates the F1 score achieved by training on instances of attribute i and testing on attribute j.

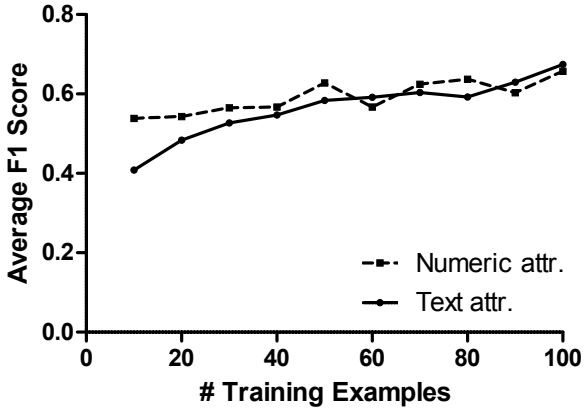


Figure 3.4: Average F1 score by number of training examples. While more training data helps, even sparse attributes reach acceptable performance.

3.6 Related Work

Large-scale extraction A popular approach to IE is supervised learning of relation-specific extractors [63]. Open IE, self-supervised learning of unlexicalized, relation-independent extractors [11], is a more scalable approach, but suffers from lower precision and recall, and doesn't canonicalize the relations. A third approach, weak supervision, performs self-supervised learning of relation-specific extractors from noisy training data, heuristically generated by matching database values to text. [38, 77] apply this technique to the biological domain, and [112] apply it to 102 relations from Freebase. LUCHS differs from these approaches in that its "database" – the set of infobox values – itself is noisy, contains many more relations, and has few instances per relation. Whereas the existing approaches focus on *syntactic* extraction patterns, LUCHS focuses on *lexical* information enhanced by dynamic lexicon learning.

Extraction from Wikipedia Wikipedia has become an interesting target for extraction. [148] build a knowledgebase from Wikipedia's semi-structured data. [162] propose a semisupervised positive-only learning technique. Although that extracts from text, its reliance on hyperlinks and other semi-structured data limits extraction. [168, 167]'s systems generate training data similar to LUCHS, but were only on a few infobox classes. In contrast, LUCHS shows that the idea scales to more than 5000 relations, but that additional techniques, such as dynamic lexicon learning, are

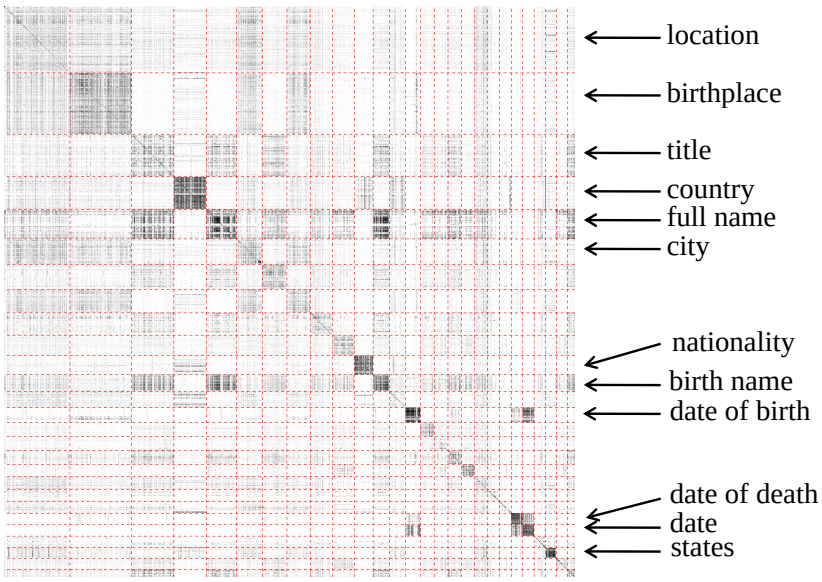


Figure 3.5: Confusion matrix for extractor accuracy training on one attribute then testing on another. Note the extraction similarity between title and full-name, as well as between dates of birth and death. Space constraints allow us to show only 1000 of LUCHS's 5025 extracted attributes, those in the largest clusters.

necessary to deal with sparsity.

Extraction with lexicons While lexicons have been commonly used for IE [34, 6, 14], many approaches assume that lexicons are clean and are supplied by a user *before* training. Other approaches [151, 111, 129] learn lexicons automatically from distributional patterns in text. [165] learns lexicons from Web lists for query tagging. LUCHS differs from these approaches in that it is not limited to a small set of well-defined relations. Rather than creating large lexicons of common entities, LUCHS attempts to efficiently instantiate a series of lexicons from a small set of seeds to bias extractors of sparse attributes. Crucual to LUCHS's different setting is also the need to avoid overfitting.

Set expansion A large amount of work has looked at automatically generating sets of related items. Starting with a set of seed terms, [56] extract lists by learning wrappers for Web pages containing those terms. [163, 164] extend the idea, computing term relatedness through a random walk algorithm that takes into account seeds, documents, wrappers and mentions. Other approaches include Bayesian methods [70] and graph label propagation algorithms [152, 16]. The goal of set

expansion techniques is to generate high precision sets of related items; hence, these techniques are evaluated based on *lexicon* precision and recall. For LUCHS, which is evaluated based on the quality of an *extractor* using the lexicons, lexicon precision is not important – as long as it does not confuse the extractor.

3.7 Conclusion

Weakly supervised learning can reduce the effort necessary for creating a relation extractor to just selecting a database of relation instances. Unfortunately, it cannot be applied for many relations due to noise and sparseness in the heuristic labeling.

This chapter showed that – with new techniques – *weakly supervised* learning of relation-specific extractors from Wikipedia infoboxes can scale to thousands of relations. In particular, we presented LUCHS, a weakly supervised system capable of learning more than an order of magnitude more relation-specific extractors than previous systems. LUCHS uses *dynamic lexicon features* that enable hyper-lexicalized extractors which cope effectively with sparse training data. We showed an overall performance of 61% F1 score, and presented experiments evaluating LUCHS's individual components.

Chapter 4

INSTAREAD: INTERACTIVELY CREATING RULE-BASED EXTRACTORS

Weak supervision enables us to learn thousands of relation extractors with minimal human effort, but it is often not applicable. The quality of extractors may be unsatisfactory for some applications, and for many relations there does not exist a database that can be used for weak supervision.

In this chapter, we propose INSTAREAD, an alternative technique that allows expert users to interactively create high-quality rule-based extractors in only 55 minutes — using no labeled training data or database. The key to reducing human effort in this technique is to make it easy for users to write rules in an expressive language and then instantly observe the impact of these rules on large corpora. To make this possible, INSTAREAD evaluates rules extremely quickly by mapping them to SQL queries and executing them in a database engine. Rules are expressed in condition-action form using logical notation, and although rules are deterministic, predicates encoding the outputs of statistical components such as a parser can be used. INSTAREAD attempts to make it even easier to write such rules by offering several accelerator tools, including a simple method for bootstrapping. Experiments on four relations demonstrate that the technique makes it possible to create high-quality extractors even when a weakly supervised approach fails.

4.1 Introduction

Weak supervision from a database allows us to learn thousands of relation extractors with minimal human effort, but it is often not applicable. Applications such as question-answering, summarization, or structured search interfaces, may require very high precision and recall, which learning with knowledge-based weak supervision may be unable to deliver. More importantly, there often does not exist a database that can be used for weak supervision. For example, Freebase does not contain the equivalent of a killed(killer, victim) relation. And even in Wikipedia only a tiny percentage of facts stated in an article are contained in its infobox.

To avoid the large amount of human effort necessary to create relation extractors today (cf.

table 1), we must therefore also consider other forms of interactions besides providing databases of relation instances. One such form is to let users directly create extraction rules. While rule-based extractors are common, little work has studied the problem of creating such extractors in the context of limited amounts of user effort. One exception is the work by Freedman et al. [62], which created extractors using a combination of manually-written extraction patterns and bootstrap learning. Starting with only a small amount of training data, development time for 5 relational extractors was confined to only one week.

We are interested in reducing development time further — to at most *55 minutes* per relation. To make this possible, we develop INSTAREAD, a novel interactive system for creating rule-based extractors that is based on three key ideas. The first key idea is that we need an expressive yet easy-to-understand rule language that allows users to leverage and connect existing components, including parsers, entity taggers, and coreference resolvers. The second key idea is that we need to provide immediate feedback to every user interaction with the system, so that users can quickly gain insights and iterate. The third key idea is that we need to equip our system with tools that make it easier for users to analyze data, and discover and write rules.

This chapter makes the following contributions:

- We present a method for interactively creating an extractor from no labeled data or database in only 55 minutes, and show how this method can be enabled by a novel system called INSTAREAD.
- We show how condition-action rules based on first-order logic can be used to tie together syntax and semantics, and how these rules can be efficiently executed on millions of documents.
- We present a set of 'accelerator' tools designed to facilitate data analysis, rule discovery and rule authoring.
- We present experiments showing that extractors created using INSTAREAD perform well, even for relations where a weakly supervised approach fails. Additional experiments analyze remaining errors, compare the effectiveness of our accelerator tools, and measure execution times.

In the next section, we discuss the problem of human effort for creating rule-based extractors in more detail. In section 4.3 we then give an overview of INSTAREAD, and the following sections discuss technical details. Section 4.7 describes experiments and section 4.8 related work. In section 4.9 we conclude.

4.2 Human Effort for Creating Rule-based Extractors

Our goal is to find ways of leveraging human effort more efficiently, so that one can create higher quality extractors in less time. Previous work has often aimed to do this by considering a particular type of feedback (such as providing a database, or labeling a given instance) and then developing algorithms for learning more accurately from such feedback. Examples are our work on weakly supervised learning described in chapters 2 and 3, work on active learning [138, 51], and work on learning with constraints [27, 69, 13, 30]. The work presented in this chapter is different in that it does not focus on enhancing an algorithm but rather the type of interaction itself. Which information should be presented to the user? How does the user provide feedback? What properties make the interaction efficient?

We study these questions in the context of an interactive system for developing rule-based extractors. Although rule-based extractors are popular, to our knowledge no work has investigated how a set of high-quality rules can be collected from a user more efficiently. In this work, we investigate this problem in more detail, but to find opportunities for improvement, we must first consider how a rule-based extractor is created today. A common development process for creating a rule-based relation extractor might look as follows.

1. **Analyze.** An expert user first tries to get a good understanding of a development corpus of text. This involves finding and analyzing example sentences containing the relation. If an existing extractor is refined, this also involves understanding the precision and recall errors that the existing extractor is making.

2. **Create Hypotheses.** After some time of analysis, the expert user formulates hypotheses about the ways that the relation is commonly expressed in the corpus and how it can possibly be disambiguated from other relations.

3. **Formulate Rules.** The expert user then translates these hypotheses into formal rules, thus creating a new extractor.

4. **Apply.** This extractor is then applied to the corpus, returning a set of extractions.

5. Go to 1.

In this development process, the expert user iteratively refines her extractor until she is satisfied with its performance on a development set. Unfortunately, each of the steps in this cycle can be very time-intensive.

1. 'Analyze' phase: The user might spend much time searching for example sentences, since in most cases only a tiny fraction of sentences contains a relation of interest. Search by keywords is sometimes possible, but relevant sentences often do not share the same keywords making it difficult to retrieve them.

2. 'Create Hypotheses' phase: Even if a user has found relevant sentences, it can take time to understand errors produced by existing rules and to develop ideas on how to avoid these. Furthermore, it can be difficult to judge which structure should be captured so that overall extraction quality is increased most.

3. 'Formulate Rules' phase: Sometimes a user understands the relevant structure that should be modeled, but is unable to translate his insights into a set of rules. This may be because the language used to represent rules is inadequate. It may be difficult to understand its semantics, it may not offer sufficient expressiveness, or it may simply not offer the necessary primitives to represent the user's ideas compactly.

4. 'Apply' phase: It may be computationally inefficient to apply a set of rules to a corpus, in which case the user is not able to observe the impact of his actions quickly. Also, many insights can only be obtained when rules are applied to large datasets, which further exacerbates the problem.

As one can see, the steps can be time-intensive for a variety of reasons. Challenges range from retrieval and visualization to formal languages and efficient computation. The cycle can be delayed for any of these reasons.

4.3 Overview of INSTAREAD

To streamline the development process for rule-based extractors, we developed INSTAREAD, an interactive development system that runs in the browser. INSTAREAD combines a variety of techniques to avoid the bottlenecks outlined above. First, its representation of rules is expressive and yet simple. Second, it is able to apply rules very quickly, even on large datasets. Third, it provides a set of user interface features designed to accelerate the steps from analysis to rule formulation. These accelerators include a method for bootstrapping (Bootstrap Tool), a method for keyword search (Keyword Tool), a method for including morphological information mined from Wiktionary (Morphology Tool), and a method for decomposing complex rules into chains of rules (Decomposition Tool).

Before we delve into the technical details, let us consider an example showing how INSTAREAD is used. We assume that our user, Anna, is interested in creating an extractor for the killed(killer,victim) relation. She starts by loading INSTAREAD in a Web browser, defining a predicate for the killed relation, and selecting a dataset for development (see Chapter 1). She then proceeds as follows:

- Anna wants to see examples of sentences containing the killed relation. She first searches for sentences containing the keyword 'killed'. In a side bar, INSTAREAD suggests to also search by distributionally similar keywords such as 'murder' and 'assassin'. With a few clicks, Anna gets an overview of many relevant sentences (Keyword Tool – Figure 4.3a).

- Trying to find common patterns, Anna checks several sentences more carefully by looking at visualizations of the syntactic structure computed by a parser and entity types computed by a named-entity tagger (Figure 4.3b). Anna soon has an idea for an extraction pattern based on these inputs.

- She clicks on the relevant parts of the visualizations to indicate relation arguments and information that should be part of the extraction pattern. The system responds by generating a list of candidate rules. She chooses one of these rules.

- Anna decides to modify the rule slightly before adding it to her extractor In particular, the rule uses an exact match for the token 'murdered', a verb in past tense, which Anna relaxes

(a) (b)

(c) (d)

Figure 4.1: Selected interactions with INSTAREAD. See example in Section 4.3.

to a match for the verb in any tense using a predicate encoding the system's knowledge about verb morphology (Morphology Tool).

- Anna now has a coarse initial extractor which she would like to refine. To find relevant rules more quickly, she now uses the system's bootstrapping method (Bootstrap Tool). The system returns a ranked list of additional rules that would extract similar tuples to those returned by her existing rules (Figures 4.3c and 4.3d). She adds several new rules to her extractor.

- After a few more minutes, Anna decides to look at the rules that she has collected so far. Looking at several rules at once, she notices that many of the rules are very similar to each

other, differing only in the stem of the verb that was used. Anna decides to refactor those rules, so that a first rule identifies a relevant verb, and a second rule identifies the syntactic structure using a relevant verb. Her new set of rules is now more compact (Decomposition Tool).

- Anna then decides to test the existing rules of her extractor more carefully. She notices that one rule produces many false positives; Anna thus tries to make it more precise by by adding another condition. After testing her extractor on a sample of sentences, Anna is satisfied with the results.

In the following sections we describe the technical details of INSTAREAD. We start by describing INSTAREAD's rule language, then discuss the four accelerator tools, and finally INSTAREAD's capability of evaluating rules extremely quickly.

4.4 Creating Relation Extractors using Logical Rules

An important factor impacting user efficiency is the space of possible inputs. With an expressive input language, feedback can be shorter and more direct, thus potentially reducing user effort. INSTAREAD accepts input in the form of condition-action rules expressed in first-order logic [1]. This has several advantages. Logical rules are relatively easy to read and write for experts. The language is generic; it can be used to model a wide variety of things, and can always be extended by defining new predicates. Our rules are similar (but not identical) to those used in logic programming languages such as Prolog, allowing us to leverage a large body of existing work. Finally, our rules can be used in more advanced statistical modeling languages, such as Markov Logic networks [48], thus making it easier to integrate statistical learning in the future.

For an illustrative example of an extraction rule, let us assume we would like to extract instances of the killed(killer,victim) relation from text. We could create the following rule:

$$\texttt{killed}(\texttt{a},\texttt{c}) \Leftarrow \texttt{next}(\texttt{a},\texttt{b}) \wedge \texttt{next}(\texttt{b},\texttt{c}) \wedge \texttt{token}(\texttt{b},\text{`}\texttt{killed}\text{'}) \wedge \texttt{capitalized}(\texttt{a}) \wedge \texttt{capitalized}(\texttt{b})$$

[1] For tractability, we require rules to be in safe domain-relational calculus [154]. See section 4.6 for details.

Here, we assume that we have defined the killed predicate as taking two arguments, and we assume that we are given predicates next (next token position in sentence), token (token at token position), and capitalized (token is capitalized). We further assume that variables a,b,c are ranging over the token positions in a sentence. If we apply this rule to the sentence 'John killed Mary last Friday.', it would extract the instance killed(John, Mary).

There are some restrictions to the space of rules: The left-hand side of each rule must consist of a single predicate with variables as its arguments, and the rule must be safe (see section 4.6). We can re-use newly defined predicates in other rules. We set a ground instance to be true only if it is implied by a chain of existing rules. For simplicity, all rules are deterministic and we execute them sequentially in a user-defined order. Although this may appear like a major limitation, we note that many state-of-the-art algorithms for tasks like dependency extraction [103], coreference resolution [123, 96], or open information extraction [58] follow this model of chaining deterministic rules.

An important question is which predicates to start with. INSTAREAD offers a set of 70 built-in predicates, like next, token, and capitalized. These predicates alone, however, would still be quite restrictive, making it cumbersome to create an extractor. Therefore, INSTAREAD also makes available predicates that encode the output of four natural language processing systems, including a phrase structure parser [28], a typed dependency extractor [103][2], a coreference resolution system [123], and a named-entity tagger [60]. This allows us to write rules that leverage parse, coreference, and entity type information simultaneously.

In figure 4.2 we show another example of a set of rules for the killed relation, defined over syntactic dependencies. Here, we first define helper relations killingNoun and killingOfVictim and then use these to define our target relation killed. Such chaining of rules allows us to represent a richer set of rules more compactly.

The rules in our examples so far have been Horn clauses, and thus did not use the full power of first-order logic. Besides conjunction ($\wedge$), we can also use disjunction ($\vee$), negation ($\neg$), and existential ($\exists$) and universal ($\forall$) quantification. While we often do not need these operators, they are sometimes convenient when we have lexical ambiguities. For example, in our evaluation dis-

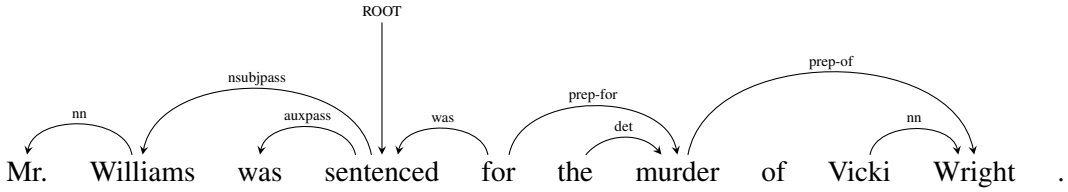
[2]We use collapsed dependencies with propagation of conjunct dependencies.

killingNoun(‘murder’)
killingNoun(‘assassination’)
killingNoun(‘killing’)
killingNoun(‘slaughter’)

killingOfVictim(c, b) ⇐ prep-of(c, b) ∧ token(c, d) ∧ killingNoun(d)
killingOfVictim(c, b) ⇐ nn(c, b) ∧ token(c, d) ∧ killingNoun(d)
killingOfVictim(c, b) ⇐ poss(c, b) ∧ token(c, d) ∧ killingNoun(d)

killed(a, b) ⇐ person(a) ∧ person(b) ∧ nsubjpass(c, a) ∧ token(c, ‘sentenced’)∧
prep-for(c, d) ∧ killingOfVictim(d, b)
killed(a, b) ⇐ person(a) ∧ person(b) ∧ prep-by(c, a) ∧ killingOfVictim(c, b)

(a) Rules. We assume that dependency predicates nn, poss, nsubjpass, , prep-by, prep-for, prep-of and named-entity predicate person have been pre-computed.



(b) Example sentence with typed dependencies.

killingOfVictim(murder-7, Wright-11)
killed(Williams-2, Wright-11)

(c) Predicted ground instances. Here, murder-7 refers to the 7th token in the example sentence, etc.

Figure 4.2: A set of rules based on dependency and named entity type information.

cussed in section 4.7, our user of INSTAREAD created the following rule to extract instances of the founded(person,organization) relation:

$$\texttt{founded}(\texttt{a},\texttt{b}) \Leftarrow \texttt{nsubj}(\texttt{c},\texttt{a}) \wedge \texttt{dobj}(\texttt{c},\texttt{b}) \wedge \texttt{token}(\texttt{c},\text{`}\texttt{built}\text{'}) \wedge \texttt{person}(\texttt{a}) \wedge \texttt{organization}(\texttt{b})$$

This rule was designed to match sentences such as: 'Michael Dell built his first company in a dorm-room.' However, this rule also incorrectly matches a number of other sentences such as: 'Mr.HarrisbuiltDellinto a formidable competitor to IBM.' While 'building an organization' typically implies a founded relation, 'building an organization into something' does not. This distinction can be captured in our rule by adding the conjunct $\neg(\exists \texttt{d} : \texttt{prep-into}(\texttt{c},\texttt{d}))$.

4.5 Accelerating Data Analysis, Rule Discovery and Rule Authoring

Although the language used to define rules is important, additional user interface components are necessary to make rule development efficient. These interface components target our identified problems in the 'Analyze', 'Create Hypotheses', and 'Formulate Rules' phases, and concern both visualization and interactions.

4.5.1 Visualization

While logical expressions are relatively easy to understand for experts, it takes time to read them. They may be long, may contain many variables or constants, or may contain unfamiliar predicates. However, for much of the information we encode in logic, there exist different, specialized representations that are easy to read and that an expert user is already familiar with.

For example, a sentence's syntactic dependencies are often represented as a graph, rather than using logical notation. INSTAREAD therefore offers the option of viewing information not only in logic, but also using such visualizations, as shown in figure 4.3.

Making rules easier to read is more challenging. Syntax highlighting may provide some benefit, but we explore a different idea. In early testing, we noticed that users would often write short comments for each rule. These comments were typically the surface pattern that corresponded to the rule. We therefore developed a technique that generated a surface pattern for each rule automatically and presented that pattern together with the rule, as shown in figure 4.4. To generate such patterns automatically, we proceed as follows. We first search for an example sentence matching the rule.

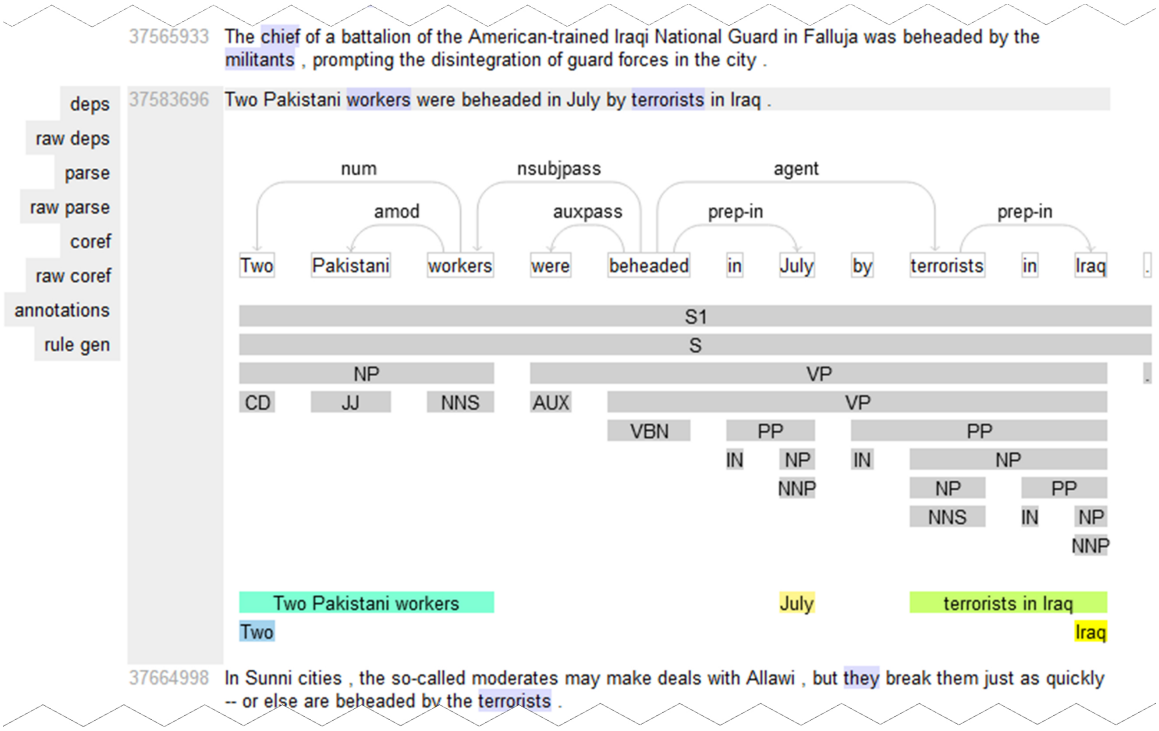


Figure 4.3: INSTAREAD can show information either in logical form or using specialized visualizations. Here, a user has activated visualizations of syntactic dependencies, phrase structure, and coreference information for a sentence.

We then analyze each rule variable that corresponds to a token or span in the sentence. Following the token order of the sentence, we then replace variables corresponding to target arguments by '...' and variables with lexical constraints by their token or span, ignoring all other variables.

Also shown in figure 4.4 is how INSTAREAD displays the number of extractions together with each rule, helping users quickly judge the importance of a rule.

Such visualizations do not convey all information encoded in the logical representation. Our goal is therefore not to replace our logical language with a visual language, but to complement one with the other. We need both the exactness of logic, and the ability to convey (approximate) meaning and relevance quickly.

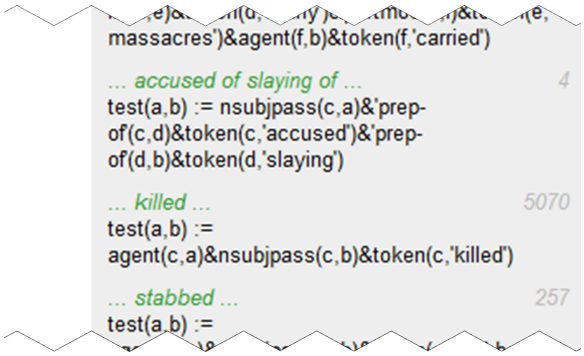


Figure 4.4: INSTAREAD shows automatically generated comments and number of extractions together with each rule, allowing users to see (approximate) meaning and relevance without needing to read logical expressions.

4.5.2 Four Accelerator Tools

While visualization mitigates the problem, rule authoring is still difficult and slow. The major problem is that it is not clear how to discover effective rules. We therefore developed four *accelerator* tools. With these tools we focus on rules which consist of conjunctions of constraints on argument entity types, syntactic dependencies, and lexical items, such as those shown in figure 4.2. Rules of this structure have been frequently used in relation extraction [112, 127, 80].

4.5.3 Bootstrap Tool

Our systems for learning extractors with knowledge-based weak supervision, discussed in chapters 2 and 3, discovered relevant features by matching tuples to sentences. With the Bootstrap tool, INSTAREAD offers a similar capability. A user can either define a set of seed tuples explicitly or implicitly through rules. These tuples are then matched to sentences; potential rules are generated, and returned back to the user for inspection.

This technique only works if the same tuples appear multiple times. To expand recall, we therefore follow [66] and also take into account coreference information. Let us assume a user has defined seed extraction rules of the form $\texttt{R}(\texttt{a}, \texttt{b}) \Leftarrow \ldots$, where `a` and `b` point to the argument head positions

of mentions of relation R. The Bootstrap tool then computes matches $\mathtt{R}^*$ as

$$\begin{aligned}\mathtt{R}^*(\mathtt{x},\mathtt{y}) \Leftarrow\ & \mathtt{R}(\mathtt{a},\mathtt{b}) \wedge \\ & \mathtt{coref}(\mathtt{a},\mathtt{a}_{\mathtt{name}}) \wedge \mathtt{a}_{\mathtt{name}} = \mathtt{x}_{\mathtt{name}} \wedge \mathtt{coref}(\mathtt{x},\mathtt{n2}) \wedge \\ & \mathtt{coref}(\mathtt{b},\mathtt{b}_{\mathtt{name}}) \wedge \mathtt{b}_{\mathtt{name}} = \mathtt{y}_{\mathtt{name}} \wedge \mathtt{coref}(\mathtt{y},\mathtt{n2}) \wedge \\ & \mathtt{pos2sentence}(\mathtt{x},\mathtt{s}) \wedge \mathtt{pos2sentence}(\mathtt{y},\mathtt{s}),\end{aligned}$$

where $\mathtt{coref}(\mathtt{c},\mathtt{c}_{\mathtt{name}})$ maps an entity mention head position c to the representative name of the entity $\mathtt{c}_{\mathtt{name}}$, and $\mathtt{pos2sentence}(\mathtt{p},\mathtt{s})$ returns the sentence s containing position p.

The tool must then generate candidate extraction patterns. For each matching pair of positions x,y, INSTAREAD generates all paths through the dependency graph connecting x and y. For each such path, it then generates a rule that is the conjunction of all dependency predicates on the path, as well as token match conditions for all tokens along the path except at x and y.

Finally, it aggregates and ranks all generated rules. INSTAREAD allows switching between two ranking scores: pointwise mutual information of a proposed rule with the user's seeds, and number of extractions a proposed rule would make. The latter may show more irrelevant queries on top, but the relevant ones have more extractions often reducing overall user effort.

4.5.4 Keywords Tool

A problem with the Bootstrap tool is that it only works if the same tuples appear multiple times in the corpus, and so its recall is limited. We therefore investigate another approach with broader coverage. This approach lets users first find relevant sentences using keyword search, and then create rules based on the patterns observed in these sentences.

INSTAREAD supports this process by automatically suggesting related keywords and their frequencies in a sidebar. Clicking on a related keyword then initiates a new keyword search returning a new set of sentences. Two types of keyword suggestions are used. The first, 'Auto Complete', simply shows the most frequent words having the user's keyword as a prefix. This allows one to see that killer is frequently used among words starting with kill. The second ranks words by distributional similarity. Specifically, we represent each word $w \in \mathcal{W}$ in our corpus as a vector of weighted words

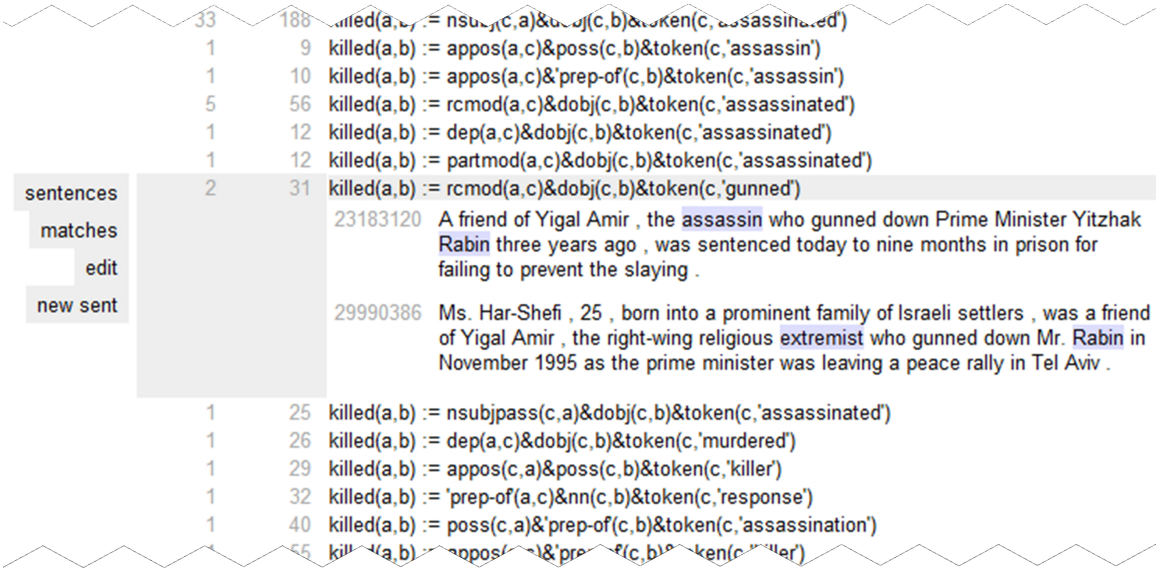


Figure 4.5: Bootstrap tool. The system has returned suggested rules for the seed rule $\texttt{a},\texttt{b} := \texttt{nsubj}(\texttt{c},\texttt{a}) \wedge \texttt{dobj}(\texttt{c},\texttt{b}) \wedge \texttt{token}(\texttt{c},\text{`}\texttt{assassinated}\text{'})$, and the user is exploring sentences which match seed tuples.

$v \in \mathcal{W}$ appearing in the sentences that w appears in, $w^0 = (w^0_{v_1} w^0_{v_2} \dots w^0_{v_{|\mathcal{W}|}})$. We then compute the distributional similarity of words $w^0, w^1 \in \mathcal{W}$ as

$$sim_{cos} := \frac{w^0 \cdot w^1}{\|w^0\|\|w^1\|}.$$

After a user has identified relevant sentences, he creates appropriate rules. Although this is facilitated with the visualizations described in section 4.5.1, we observed that this could still take a long time. A user would typically inspect the sentence's dependency graphs, find a path between the arguments, and then write out a logical expression. To accelerate this almost mechanical task, we added a small feature to the user interface, that allowed users to semi-automatically generate rules: Double-click on a token identified that as an argument, and single click indicated that a token match condition should be added. After each interaction, the system computed possible dependency paths and returned a set of candidate rules which could be further edited and collected. In informal tests, this feature reduced rule authoring time by about half.

4.5.5 Morphology Tool

Unfortunately, a large amount of user effort is often spent on encoding simple syntactic variations which follow well-known grammatical rules. Since such rules are typically relation-independent, we would like to pre-populate the system with these rules in order to reduce user effort.

In a first step, we encoded a subset of grammatical rules. Given a verb base form, tense, voice, and person, we can generate a logical rule that finds all instances based on dependencies and lexical items. Note however, that to make these mappings we need to know about English verb inflections. We mined verb inflections from Wiktionary. An alternative, potentially lower precision, approach would be to use a stemmer. INSTAREAD makes these syntactic rules available with new predicates `actInd` and `passInd` for finding all mentions of active and passive indicatives of a verb. For example, `actInd`(`subjPos`, `verbPos`, '`shoot`') matches 'He shot John' as well as 'He used to shoot John'.

4.5.6 Decomposition Tool

Different rules for a relation extractor often share similar subparts. In such cases, it makes sense to define new predicates for these subparts, and then re-use these predicates in other rules, as shown in our example in figure 4.2. Such decomposition not only makes the set of rules more compact; it usually also expands recall because many valid combinations of subparts may not be directly observed in the development set.

INSTAREAD makes it easy to define new predicates. These new predicates are immediately available for use in rules. At the moment, decomposition is performed by manually editing rules. In the future we would like to automatically suggest possible decompositions using clustering.

4.6 Efficient Rule Evaluation with a Database

INSTAREAD depends on the ability to execute complex rules over large amounts of text *instantly*. To enable this capability, we use a database management system. In this section, we briefly summarize how we map data and rules to a database.

Data Types Each predicate has a signature such as `name(type1, type2)`, that includes the name of the predicate as well as the data types of its arguments. Data types indicate how arguments are physically represented and are a different concept from named-entity types. INSTAREAD offers data types `Int`, `Pos`, `Span`, `Str`, and `Obj`. For each, there exists a defined mapping to a vector of SQL data types. For example, the type to represent a token position, `Pos`, is mapped to `[Integer, Byte]` for storing a sentence identifier and a token offset.

Predicates Most predicates are *extensional*, i.e. the instances of the relation are stored directly in a relational table. Some predicates, however, are *intensional*, i.e. defined only by a query over different relations. An example of an intensional predicate is `str2span(str, sp)` which returns all mentions of a multi-word string (figure 4.6). Each user-defined extensional predicate is mapped to two relational tables, a data and a provenance table. Whereas the data table contains all tuples of the predicate, the provenance table keeps track of the rules or user actions that generated an entry in the data table, and is used to update the data table when a rule is deleted. A small set of pre-defined extensional predicates contain an enormous number of tuples. For such predicates, INSTAREAD uses more compact representations.

Rules The key component of INSTAREAD's implementation is a translation of logical rules into SQL queries. SQL, however, does not support arbitrary logical expressions. To ensure that answers are sensible, for example have no infinite result sets, SQL is limited to expressions that satisfy certain syntactic criteria. SQL[3] is based on relational algebra which is equivalent in expressive power to safe domain-relational calculus [154], the subset of logic used by INSTAREAD. INSTAREAD first parses rules into an abstract syntax tree representation of first-order logic, then checks for safety, performs type inference and linking, then translates into tuple relational calculus, and eventually – using our predicate and type mappings – into SQL.

Performance For user-defined extensional predicates, INSTAREAD maintains one index for each column. Pre-defined extensional predicates use different multi-column indices. A variety of information is pre-computed, including phrase structure trees, dependencies, coreference information,

[3] In addition to relational algebra, SQL supports aggregation operators, and some commercial implementations support recursion, making SQL Turing-complete.

$\texttt{r}(\texttt{t}) \Leftarrow \texttt{str2span}(\text{`Lee Harvey Oswald'}, \texttt{s}) \wedge$
$\texttt{span2pos}(\texttt{s}, \texttt{p}) \wedge \texttt{nsubj}(\texttt{c}, \texttt{p}) \wedge$
$\texttt{token}(\texttt{c}, \texttt{t})$

$\xrightarrow{\text{translation}}$

```
SELECT ti4.tokenID
FROM tokenInst ti0, tokenInst ti1,
  tokenInst ti2, tokenInst ti3,
  dependencyInst di0, tokenInst ti4
WHERE ti4.offset = di0.from
  AND ti4.sentenceID = di0.sentenceID
  AND di0.to = ti3.offset
  AND di0.sentenceID = ti3.sentenceID
  AND di0.dependencyID = 11
  AND ti3.offset < ti2.offset + 1
  AND ti3.offset >= ti0.offset
  AND ti3.sentenceID = ti0.sentenceID
  AND ti2.tokenID = 79216
  AND ti1.tokenID = 6058
  AND ti0.tokenID = 5322
  AND ti0.offset + 2 = ti2.offset
  AND ti0.sentenceID = ti2.sentenceID
  AND ti0.offset + 1 = ti1.offset
  AND ti0.sentenceID = ti1.sentenceID
```

Figure 4.6: Translation of a (safe) expression in first-order logic to SQL. The expression returns verbs for which Lee Harvey Oswald appears as subject. `str2spans` and `span2pos` are intensional predicates, `nsubj` and `token` are extensional.

named-entities, rule candidates for Bootstrap tool, distributional similarity for Keyword tool, and verb mentions for Morphology tool.

INSTAREAD uses a MySQL database. In practice, the size of the indexes often leads to poor locality in disk access patterns, so we deployed the system on a PCI express-based solid state drive.

4.7 Experiments

We performed several experiments to evaluate the effectiveness of INSTAREAD. In particular, we were interested in the quality of relational extractors that an expert could create in less than 1 hour. We were also interested in comparing the effectiveness of the four accelerator tools presented in section 4.5.2, and in measuring rule execution times. Finally, we were interested in the types of errors that the relational extractors made, and how these could be reduced.

For our evaluation we used two datasets: **NYTimes.** The New York Times Annotated Corpus [136] contains 1.8M new articles published between 1987 and 2007. **CoNLL04.** This corpus, created by Roth and Yih [132], contains 5516 sentences from news articles with annotations for several relations.

We selected four relations for which our weakly supervised system, MULTIR, did not perform so well: attendedSchool(person,school), founded(founder,organization), killed(killer,victim), and married(spouse1,spouse2). These relations were also selected because they cover a range

of domains, they are important in many information extraction evaluations [85, 47, 132], they are binary, and they do not depend on recognizing uncommon entity types. We would like to develop tools to create recognizers for other less common entity types in future work. We used the Stanford NER system for identifying named entities of the types person and organization. In the case of attendedSchool we additionally created a recognizer for type school by listing 30 common head words such as 'University' or 'Academy'. This process took under 5 minutes. We then split the NYTimes corpus randomly into a development set and a test set. Each of these sets contained about 22M sentences.

During the evaluation, an expert user applied INSTAREAD on the NYTimes development set to create a relational extractor for each of the four relations. For each relation the user was stopped after 55 minutes. To enable a comparison of the impact of each accelerator tool, the user was also required to use the tools in a sequence with a fixed amount of time per tool. For the Bootstrap tool 15 minutes, for the Keyword tool 15 minutes, for the Morphology tool 5 minutes, and for the Decomposition tool 20 minutes.

An extraction consisted of a pair strings representing named entities as well as references to their mentions in a sentence that expressed their relation. For the CoNLL04 dataset, we used the existing annotations to determine the correctness of an extraction. To measure precision on the NYTimes dataset, we sampled 100 extractions and manually created annotations by following the ACE annotation guidelines [47]. Precision and recall were computed on a mention level.

During the experiment, we logged all user interactions with the interface as well as SQL queries that were executed on the database.

4.7.1 Extraction Quality

		attendedSchool	founded	killed	married
Rules	Precision	1.00	.91	.90	.90
	# extractions	1,411	997	189	4,694
Weakly	Precision	0	.71	N/A	.50
supervised	# extractions	5	14	N/A	2

Table 4.1: Precision and number of extractions on the NYTimes07 test set. We selected four relations for which our weakly supervised system, MULTIR, did not perform well.

	attendedSchool	founded	killed	married
False Positives				
NER	0	6	1	0
dependencies	0	1	5	10
rules	0	2	4	0

Table 4.2: Precision errors on a sample of 100 sentences from an independent test set from the NYTimes.

We first analyze the overall quality of the extractors. Table 4.1 summarizes results on the NYTimes07 test dataset. As shown in the table, our weakly supervised system, MULTIR did not perform well on these relations. In contrast, INSTAREAD's precision was 90% or higher for all four relations, and each extractor returned hundreds of tuples. For relation founded, MULTIR's estimated precision of 71.4% compares to 91% for INSTAREAD. More interesting, however, is the number of sentence-level extractions. MULTIR only identified 14 mentions of the founded relation in its test set, but for the same sentences INSTAREAD returned 997 — a 71x increase. The two systems use the same NER tagger, but INSTAREAD also uses coreference resolution. Without coreference resolution INSTAREAD returns 617 mentions. We further note that more generally knowledge-based weak supervision cannot be applied for a large number of relations, simply because there does not exist an appropriate database. For example, at the time of writing there does not exist a mapping for the killed relation to relations in Freebase.

We then investigated INSTAREAD's remaining precision errors (Table 4.2). For relation attendedSchool no precision errors were found in our sample, while the errors of the other three relations fall into three groups: errors caused by incorrect dependency extractions (16), errors caused by incorrect NER extractions (7), and errors caused by overly general rules (6). Only the last group of errors are caused by the user of INSTAREAD. All were due to double meanings of the words fell, executor, and built (see section 4.4).

In order to estimate recall of INSTAREAD's extractors, we apply these to the CoNLL04 dataset, for which we have gold annotations. For the killed relation, we obtain a recall of 34.2% at a precision of 97.8%, giving us an F1 score of 50.7%. Table 4.3 shows a breakdown of the errors. About 36% of errors are due to problems during preprocessing, such as named-entity recognition, dependency extraction, and coreference resolution. The remaining 64% are due to insufficient coverage of the

False Negatives		False Positives	
Preprocessing (missing predictions)		Preprocessing (wrong predictions)	
NER	30	NER	0
dependencies	25	dependencies	2
coreferences	7	coreferences	0
Rules (missing predictions)		Rules (wrong predictions)	
lexical items	89	lexical items	0
syntactic variation	17	syntactic variation	0
reasoning chain	10	reasoning chain	0

Table 4.3: Error Analysis on CONLL dataset for killing relation. More than one third of errors are caused by incorrect preprocessing.

created set of rules. The vast majority of these involve lexical items for which no rule has yet been defined. Others involve syntactic variations of existing rules. A small number would require more complex reasoning which we do not consider at this point. We believe that rule coverage would be better if our user is given a longer period of development time, since the number of distinct lexical items that are missing is relatively small. Only eight new rules would cover more than half of all errors. These include phrases such as 'to act in a killing', 'to fire the shots that kill', 'to plead guilty in a killing', and 'to be killed by group that includes ...'. Also, many missing syntactic variations could be obtained by further decomposing existing rules.

We also compare our results to supervised extraction. Roth and Yih [132] report a recall of 81.3% at a precision of 82.2% for an F1 score of 81.7% — as far as we know the best reported results for this dataset. In their experiments they assume gold annotations of argument segments. With gold argument segments, INSTAREAD's extractor delivers an F1 score of 64.8%. Using an enhancement that will be described in section 4.7.4, the F1 score climbs to 70.6%. While this number is below the one reported by Roth and Yih, note that it did not use thousands of manually annotated sentences, thus requiring far less human effort.

4.7.2 Comparison of Accelerator Tools

Figure 4.7 shows the contribution of each interface tool to the number of extractions. The vast majority of extractions, 84%, were obtained by rules created during the Bootstrap phase. The Bootstrap tool has the ability to aggregate over many potential rules and then rank those taking into account

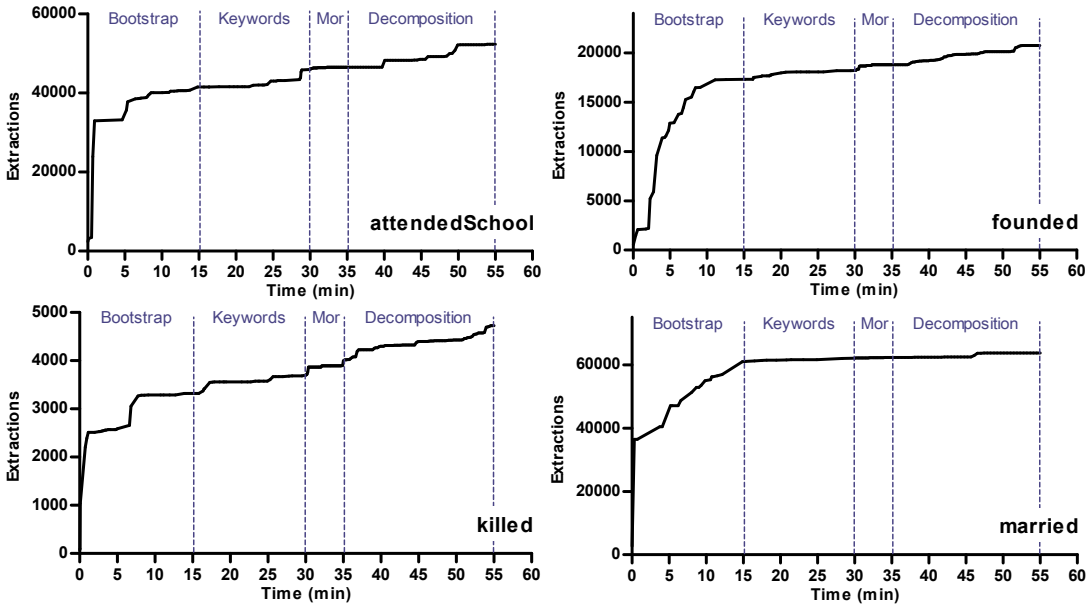


Figure 4.7: Increase in number of extractions on an independent test set when using INSTAREAD for 55 minutes. The Bootstrap tool allows to capture a large number of extractions quickly, but often does not yield additional gains after a few minutes. The Keyword tool enables slow, but consistent gains. The Morphology tool provides a small gain. Decomposition is helpful when there exist a large number of lexical stems that imply a relation (e.g. for the killed relation).

the number of extractions. This ranking ensures that user effort is directed to rules which are likely to matter most. Such ranking is not possible with the Keyword tool, which, however, has a different advantage: It can find rarely used ways of expressing a relation. In contrast, the Bootstrap tool only works if the same relation instance is expressed multiple times in different ways. We therefore often observe that it provides no more improvement after a few minutes of use. 3.4% of extractions were obtained by rules created during the Keyword phase, 2.6% during the Morphology phase, and 9.5% during the Decomposition phase.

4.7.3 Runtime Performance

INSTAREAD depends on the ability to execute complex rules instantly, so we also investigate runtime performance. The time of translating from our first-order logic input language to SQL was negligible, so we focus on SQL query execution time. The development database contained 22M

sentences, a total of 3.7B rows in 75 tables, and used about 140GB of disk space. Table 4.4 presents key metrics of the SQL queries executed by our user during rule development time. The majority of queries executed in 74ms or less. A small number of outliers, however, took significantly longer. These long-running queries, which in the worst case took 52.5 seconds, were initiated by the Bootstrap tool. These queries first identified all matches of a user-defined seed rule, then for each match determined the coreference clusters of the arguments, then retrieved other sentences that referenced the same arguments, then retrieved dependency paths connecting those references, and finally aggregated and ranked these dependency paths. One problem with these queries is that the ranking phase at the end is only possible after all intermediate result sets have been computed, and these intermediate result sets can be very large. While we see no more opportunities for speedups through indexing and pre-processing, we would like to explore approximate computations using sampling in the future.

	avg	median	max
# SQL queries per relation (55min)	55	54	66
# join tables per SQL query	4.3	4	11
execution time (s) per SQL query	1.5	.074	52.5

Table 4.4: Metrics of database queries executed during rule development time. On 22M sentences, the majority of queries execute in 74ms or less.

4.7.4 Towards Joint Parsing and Relation Prediction

Our error analyses in tables 4.2 and 4.3 show that a large number of extraction errors are caused by incorrect preprocessing. For example, at least 15% of errors were caused by wrong syntactic dependencies. In our experiments, we used the Stanford dependency extractor to compute dependencies from phrase-structure trees, and we used the CJ parser to compute these trees from text. Our investigation revealed that in most cases the errors were already caused by incorrectly parsed trees.

Such errors might be avoided if inference during preprocessing and relation extraction were performed *jointly*, but many existing approaches to joint-inference are computationally expensive and difficult to scale. INSTAREAD's deterministic high-precision rules, however, may provide a simple and effective alternative method. In particular, we propose to choose between different can-

didate predictions during preprocessing by considering basic properties of the relation extractions that would be obtained for each candidate prediction.

For joint parsing and relation prediction this idea can be implemented as follows: For each sentence, the CJ parser can output not only the best parse, but a ranked list of k parses where k is configurable parameter. Our analysis showed that when the top dependency parse was incorrect, in 35% of cases the correct parse was at position 2, in 50% of cases among the top 5, and in 90% among the top 50. In a preliminary experiment, we heuristically selected a parse for each sentence as follows: We first determined the set of parses among the top 50 that would lead to the maximum number of relation extractions, and then returned the highest-ranked parse among this set. Table 4.5 shows the impact of this approach on the CoNLL04 dataset and our rules for the killed relation. Recall expanded by more than 20% at virtually no loss in precision.

	Recall	Precision	F1
INSTAREAD	.342	.978	.507
INSTAREAD-joint	.412	.973	.58

Table 4.5: Impact of joint parsing and relation prediction for the killed relation on the CoNLL dataset.

It is not surprising that recall improved, given that the correct parse is frequently among the top 50, — but why is there no drop in precision? The extraction rules depend not only on syntactic dependencies, but also require matches on entity types and lexical items which makes co-incidental matches unlikely.

4.7.5 Discussion

Our experiments provided many valuable insights beyond the numbers presented above, some pointing towards promising opportunities for future improvement. Just like the ideas discussed in section 4.5, these too might impact development time significantly.

First, rules were manually edited during the Morphology and Bootstrap phases, but these transformations could be almost entirely automated. This alone could reduce development time by 25 minutes per relation. Since these phases were sometimes aborted before our user decided to be

done, automation may even deliver higher recall. Recall could be further increased by extending the Morphology tool to handle additional syntactic variations, for example involving participle phrases.

While the Bootstrap tool is particularly effective, its recall is limited. We already mitigated this problem by integrating coreference information, but we could expand recall further by also integrating our (automated) Morphology and Decomposition tools. Interestingly, this might also change the nature of the rules suggested by the Bootstrap tool. Since the integrated components would cover many syntactic variations, the remaining suggestions might look more semantic (for example synonymy of words), and there would be fewer of them. Since our matching currently only considers exact matches, we could additionally expand recall by adding a model for named-entity linking.

We also noticed a number of subtle shortcomings in INSTAREAD's user interface. One such shortcoming is that the system did not hide sentences in the Keyword tool which had already triggered a rule. Similary, it did not hide rules in the Bootstrap tool which had already been explored. Hiding such sentences and rules may allow users to find relevant ones more quickly.

The quality of our relational extractors depends on the quality of external components for named-entity extraction, dependency extraction, and coreference resolution. Our experiments in section 4.7.4 show the promise of a joint inference approach. However, we also encountered cases which motivated modifications to these external components. For example, we noticed that the Stanford dependency representation was missing dependencies necessary for relation extraction. It did not encode the prepositions used when a relative clause started with a preposition. This made it impossible to distinguish certain cases. To facilitate such changes, we would like to extend INSTAREAD so that these components can be developed using similar a similar technique. We believe that the deterministic, rule-based architecture of the Stanford Dependency extractor and Coreference resolution system would facilitate such an integration.

The expressiveness of the rule language was important since it enabled the Morphology and Decomposition tools, both of which improved results. The logical operators $\neg$, $\exists$, and $\forall$, however, were used rarely in the experiments. They appear to be less important for relation extraction. Yet our inspection of rules used in the Stanford Coreference system suggests that they will be important for the proposed deeper integration.

4.8 Related Work

Freedman et al. [62] report results of an evaluation that created a question answering system for a new domain in one week. One finding was that hand-written rules outperformed a system based on bootstrap learning, but that their combination was best. The bootstrap learning system, unlike INSTAREAD's Bootstrap tool, ran autonomously without user interaction. While their evaluation included creation of relational extractors, named-entity extractors, and a coreference resolution system, we focus on relation extraction and explore the effectiveness of a different set of tools.

A large amount of other work has looked at bootstrapping extractors from a small set of seed examples. Agichtein & Gravano [7], Ravichandran & Hovy [125], and Pantel & Pennacchiotti [117] propose approaches for generating surface patterns for relation extraction. While these take a small set of seed examples as input, they are not evaluated in an interactive setting where a user makes iterative improvements.

Much work has also tried to make learning algorithms more accurate with different types of user input. One popular is active learning. Miller et al. [111] learn a Perceptron model for named-entity extraction; unlabeled examples are ranked by difference in Perceptron score. Riloff [130] proposes an approach to named-entity extraction, which requires users to first classify documents by domain, and then generates and ranks candidate extraction patterns by a product of log frequency and probability of the domain given the pattern. Active learning has also been studied in more general contexts, for learning probabilistic models with labeled instances [153] or labeled features [51]. Besides active learning, researchers have also tried to make learning algorithms more accurate when these are supplied additional constraints [27, 69, 13, 30]. Approaches based on active learning and learning with constraints are different from INSTAREAD in at least two ways. First, they have not been evaluated on relation extraction tasks. Second, and more importantly, their general approach is to consider a particular type of feedback and then develop algorithms for learning more accurately from such feedback. In contrast, our approach is not to compare algorithms, but to compare different types of feedback and presentation.

Database research has looked at scaling rule-based information extraction with declarative languages and query optimization. Shen et al. [139] propose using declarative Datalog with user-created predicates that execute procedural code. Krishnamurthy et al. [90, 32] propose a new lan-

guage similar to SQL. Going beyond rules, Wang et al. [160, 159] propose to also specify statistical models declaratively and then execute them in the database system. In that and other work [126, 161] the key idea is to find the optimal rule execution plan based on the statistics of the data. INSTAREAD does not use extraction-specific query optimizations, but benefits from standard SQL query optimization. Unlike our work, this line of database research does not evaluate the effectiveness of these languages with users; nor does it evaluate extraction quality.

In research on human-computer interaction, spreadsheet software has been a major success story. Spreadsheet software is similar to INSTAREAD in the sense that it allows users to quickly explore and manipulate datasets. Users can even create 'rules' which for spreadsheets are called formulas. Formulas can use built-in functions and values in existing cells to compute values for new cells. However, spreadsheet software is limited to tabular data, typically in numeric and ordinal form, wheras INSTAREAD has been designed for extracting knowledge from unstructured text.

4.9 Conclusion

Weak supervision allows creating extractors with minimal human effort, but unfortunately it is often not applicable. Sometimes the quality of an extractor is not acceptable, other times there is no database that can be used for weak supervision.

This chapter presents INSTAREAD, a system and method for creating an extractor interactively using no labeled data or database in only 55 minutes. INSTAREAD is designed to reduce human effort by allowing experts to write extraction rules in a simple, yet expressive rule language based on first-order logic, and then instantly analyze their behavior on a large text corpus. To further accelerate rule writing, INSTAREAD also provides user interface components to facilitate data analysis, rule discovery, and rule authoring. Our experiments demonstrate that extractors created using INSTAREAD can significantly outperform those learned by a weakly supervised approach.

Chapter 5

SMARTWIKI: SYNERGISTIC PAIRING OF WEAKLY SUPERVISED EXTRACTION AND ONLINE COMMUNITIES

The interactions we have discussed so far have been designed to enable experts to efficiently create relation extractors. Our weakly supervised approaches, for example, do not require more effort than selecting a database of relation instances. Unfortunately, the quality of weakly supervised extractors is sometimes unsatisfactory, and it may be expensive to collect additional feedback from experts.

In this chapter, we are therefore interested in collecting feedback from non-experts. More specifically, we explore an approach that attempts to create a synergy of weakly supervised relation extraction with content creation by the community of visitors to Wikipedia articles. After developing and refining a set of interactions to present the verification of extractions as a non primary task in the context of Wikipedia articles, we develop an innovative use of Web search advertising services to study people engaged in some other primary task. We demonstrate our proposed synergy by analyzing our deployment from two complementary perspectives: (1) we show we *accelerate community content creation* by using relation extraction to significantly increase the likelihood that a person visiting a Wikipedia article as a part of some other primary task will spontaneously choose to help improve the articles infobox, and (2) we show we *accelerate relation extraction* by using contributions collected from people interacting with our designs to significantly improve extraction performance.

5.1 Introduction

The explosion of information available on the Web presents important human computer interaction challenges. Many techniques developed to address these challenges leverage the *structure* of Web content. For example, faceted browsing exploits a set of attribute/value pairs for objects in a collection [172]. Browser enhancements like Sifter parse structured content, such as product search

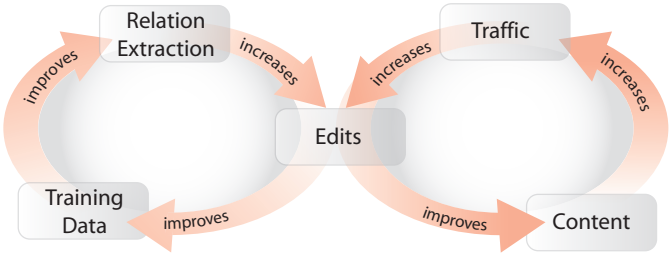


Figure 5.1: We envision the synergistic pairing of relation extraction with community content creation, using the same edits to accelerate both feedback cycles.

results, to enable interactive sorting and querying [84]. Bibliographic sites like Citeseer locate and parse citations, enabling reference counting and navigation among related documents [72]. Web search interfaces like Assieme identify and leverage relationships among and within Web pages to better support common search tasks [79]. Despite the differing goals of this variety of systems, a fundamental challenge underlies all such systems: *How can systems scalably obtain the necessary structured information?*

One popular approach that we have been discussing so far is *relation extraction*. Sifter, for example, uses a set of heuristics to identify typical patterns, such as page numbers in search results [84]. Systems like Citeseer and Assieme use a combination of heuristics and statistical machine learning algorithms [72, 79]. Although learning-based approaches can be powerful and robust, they have at least two important limitations. First, supervised learning algorithms require numerous labeled training examples, whose collection is typically expensive and time consuming. Second, learning methods, especially weakly supervised ones, can be error prone, and state of the art systems with precisions of 80 to 90% are considered successes. Although this performance can enable many applications, it is unacceptable for Wikipedia and many other sites.

A second popular approach to structured information is *community content creation*. People visiting the photo sharing site Flickr or the social bookmarking site Delicious, for example, can browse photos and bookmarks using tags applied by other people. Amazon and Netflix provide recommendations based on community contributed ratings. Finally, Wikipedia is well known for its community created articles. Despite such examples of extremely successful community approaches, many other sites have been unable to bootstrap themselves to critical mass or to overcome work/benefit

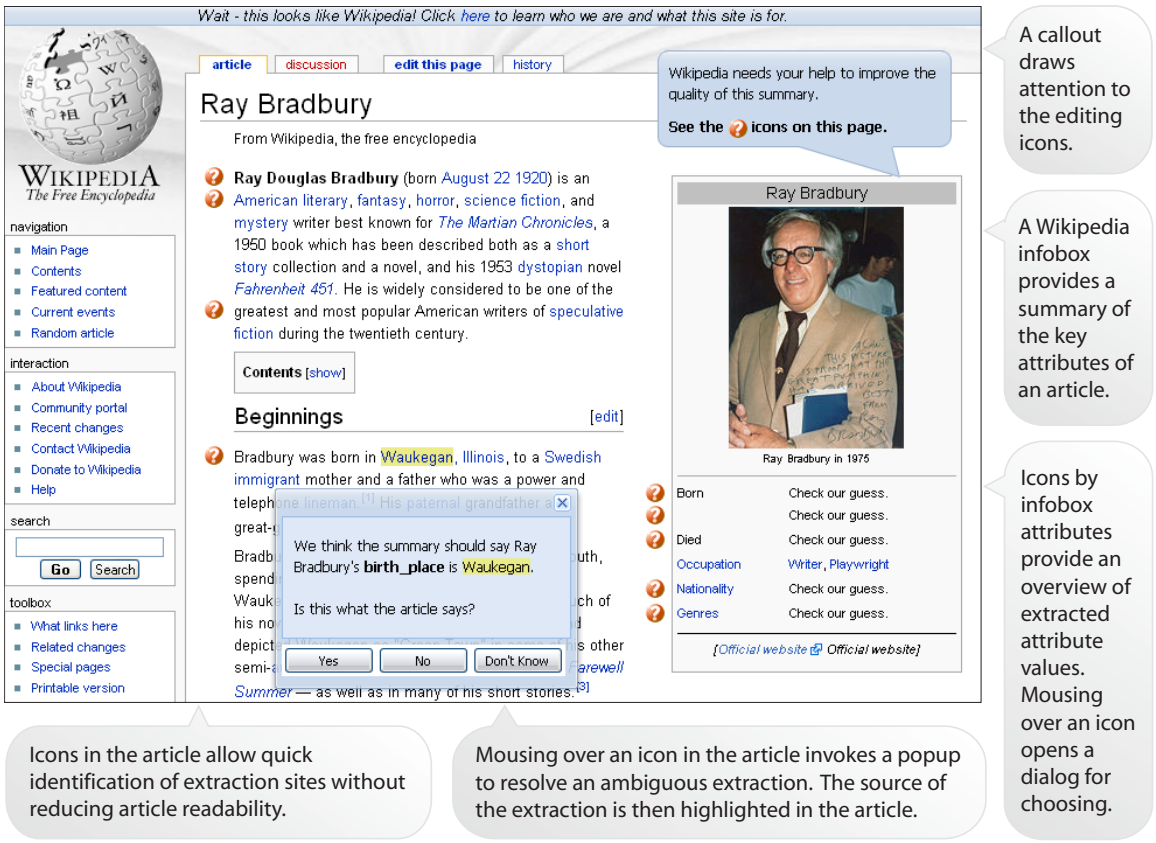


Figure 5.2: An example page containing several opportunities for mixed-initiative contribution to Wikipedia. The person viewing this page has moused over an icon in the page that indicates that the system has analyzed the text of the article and found a potential value for Ray Bradburys birthplace. The persons response to this question will be used to improve both the relation extraction system and the content of this page.

disparities [6]. Significant research has therefore explored how and why people contribute to sites like Wikipedia [19, 91, 122, 158]. This research has shown that the vast majority of work is usually done by a relatively small set of people. We are therefore interested in new methods for lowering barriers to editing and for incenting broader contribution.

Existing work has explored both relation extraction and community content creation, but has focused on these approaches in *isolation*. In contrast, we see the greatest leverage in the *synergistic pairing* of these two approaches. We envision a pair of interlocking feedback cycles, illustrated in figure 5.1. The left cycle corresponds to the traditional training of a relation extraction system,

wherein a person manually annotates a corpus with labels. After learning and testing an extractor, a person can examine the results and provide additional data to improve performance. Similarly, the right cycle corresponds to the familiar bootstrapping problem in community content creation, wherein quality content is required in order to attract people so that they might further contribute.

This paper explores the great potential synergy promised if these cycles can be made to *accelerate each other* by exploiting the same edits to advance both learning-based relation extraction and community content creation. This synergy might enable many benefits, such as the semi automated maintenance of portions of community content sites, the bootstrapping of new sites with knowledge extracted from the larger Web, and even the eventual semantification of much of the existing Web. Realizing this synergy requires new designs that both (1) leverage relation extraction to increase visitor contribution rate, and (2) leverage visitor contributions to improve the reliability of relation extraction.

We explore these challenges in the context of Wikipedia and the *Kylin* information extraction system [167, 168]. More specifically, we focus on Wikipedia infoboxes, tabular summaries present in many Wikipedia articles (figure 5.2). Having an architecture similar to LUCHS, Kylin analyzes Wikipedia articles containing infoboxes and learns to extract values from untagged articles using knowledge-based weak supervision. Although the details of our current work are tuned for Wikipedia, we argue that our synergistic approach is potentially relevant to many different types of websites.

This chapter makes the following contributions:

- We identify the potential for synergistically pairing community content creation with learning based relation extraction, using the same edits so that both feedback cycles accelerate each other.
- Using the Wikipedia community as a case study, we examine the challenge of simultaneously addressing the needs and norms of both learning based relation extraction systems and social communities.
- We develop and refine a set of interfaces to present the verification of Kylin extractions as a non primary task in the context of Wikipedia articles. Our designs leverage Kylin extractions to make it possible to contribute to improving a Wikipedia article with just a few mouse clicks, and we develop several designs to explore a tradeoff between contribution rate and unobtrusiveness.

- We develop an innovative use of Web search advertising services to study people interacting with our interfaces while engaged in some other primary task (the task that prompted them to perform the Web search that eventually brought them to our page).
- We demonstrate our desired synergy through a pair of complementary analyses: (1) we show we *accelerate community content creation* by using Kylins relation extraction to significantly increase the likelihood that a person visiting a Wikipedia article as a part of some other primary task will spontaneously choose to help improve the articles infobox, and (2) we show we *accelerate relation extraction* by using contributions collected from people interacting with our designs to significantly improve Kylins extraction performance.

5.2 Method

We first interviewed three senior members of the Wikipedia community (two administrators and a veteran contributor, all of whom had been contributing for at least four years), meeting face to face with each for approximately two hours.

Given our interest in using the same edits to drive both feedback cycles, our interviews focused on why people do or do not contribute to Wikipedia, what aspects of the Wikipedia community are difficult for newcomers, and the role a system like Kylin could play in Wikipedia.

Informed by prior work on interruptions and ambiguity resolution [40, 102, 109, 140], we next designed three interfaces examining different ways to leverage Kylin's relation extraction to accelerate community content creation. Our designs share a focus on promoting ambiguity resolution as a *non-primary* task, but they explicitly probe the tradeoff between contribution rate and unobtrusiveness.

We refined and informally evaluated our designs in informal talk aloud sessions. We presented them to nine participants (randomizing order to address potential carryover effects) and asked them to comment on aspects of the interaction they found difficult, discuss aspects of the interface they found obtrusive, and to provide overall indications of their preference. Because our goal was to refine the designs, we made improvements throughout these sessions.

Because it is difficult to envision a laboratory study which measures how often people spontaneously contribute to Wikipedia, we evaluated our synergistic approach through a novel use of Web

search advertising services. By placing ads for 2000 Wikipedia articles, we attracted visitors who were engaged in some other primary task. We assigned these visitors to different study conditions, logged their interaction with our designs, and examined their contribution rate.

5.3 Designing for the Wikipedia Community

Our interviews with veteran Wikipedia contributors, together with prior work examining the Wikipedia community [19, 91, 122], helped us identify two critical constraints governing the integration of relation extraction into Wikipedia: (1) a need to balance Wikipedia policy regarding *bots* with policy that contributors *be bold*, and (2) the opportunity to encourage greater participation in Wikipedia. Taken together, these have led us to pursue a mixed initiative approach using interfaces designed to solicit contribution as a non primary task.

5.3.1 Being Bold, Bots, and a Mixed-Initiative Approach

Wikipedia policy states that people should be bold when updating pages [3]. This policy recognizes that some edits are contentious and must wait for discussion to yield consensus, but that Wikipedia develops faster when more people contribute. It is therefore important that people be bold enough to make edits. The policy emphasizes, for example, that people should feel comfortable correcting copy editing mistakes and factual errors, rather than flagging content for discussion or for others to correct.

Wikipedia also has an explicit policy regarding automated *bots* [4], including the requirement that they be both "harmless and useful". At the time of this writing, Wikipedia lists 392 approved bots. These perform such tasks as updating links between different language versions of Wikipedia, maintaining lists, and archiving old discussion pages. Bots are appropriate for this work because simple programs provide error-free performance and because the automation frees members of the community to do other work. In contrast, it is clear Kylin should not autonomously add infobox values, because its precision currently ranges between 75 to 98% [167, 168] and the errors associated with this state of the art performance would likely be considered harmful.

One approach would be to automatically post Kylin extractions on article *talk* pages, hoping people will manually make the necessary edits (each Wikipedia article has an associated *talk* page

where changes to the article can be discussed). Although this would ensure extraction errors are not automatically introduced into Wikipedia infoboxes, the compatibility of this approach with the spirit of *be bold* is less clear. Updating an infobox with a birthdate that already appears in the body of an article is not likely to be contentious or require consensus. Instead, it is simply important that the extraction be confirmed as correct.

In addition to being a poor match to Wikipedia's *be bold* policy, posting extractions to article talk pages also fails to enable our desired synergistic pairing. In order to advance the relation extraction feedback cycle, a system needs to collect additional labels by learning whether extractions are correct. Even if a system monitored changes to an article in order to observe whether a talk page suggestion was eventually enacted, it is not clear how to interpret edits. For example, a page might change significantly between the time an extraction was posted and the time the infobox is edited. Furthermore, a person might update an infobox in a manner similar, but not identical to, the suggested edit. In these and many more situations, it is unclear what relationship an edit might have to a posted extraction.

We therefore focus on a mixed initiative approach [82], wherein potential infobox contributions are automatically extracted but then manually examined and explicitly confirmed before being published. This addresses all of the challenges discussed above. We enable the relation extraction feedback cycle with additional training data collected through explicit indications of whether an extraction is correct. We address the requirement that *bots* be harmless with the manual confirmation of Kylin extractions, and we address the spirit of the *be bold* policy by designing interfaces to present the confirmation of Kylin extractions as a non primary task in the context of Wikipedia articles, as discussed next.

5.3.2 Contribution as a Non Primary Task

Any community content creation system must provide an incentive for people to contribute, and there are many ways one might incent people to examine and confirm Kylin extractions. For example, Doan et al. propose requiring people provide a small amount of work before gaining full access to a service [106]. We believe, however, that coercive approaches are unacceptable to the Wikipedia community, whose culture is based in altruism and indirect author recognition [19, 91, 122]. Exis-

ting systems, such as the AutoWikiBrowser [2] and Cosley et al.'s SuggestBot [37], focus on experienced Wikipedia contributors who are already motivated to contribute, helping them to find work.

Instead of targeting experienced Wikipedia contributors (perhaps by posting links in article talk pages that bring experienced contributors to a page where they could explicitly confirm mixed initiative extractions), we believe our desired synergistic pairing is better served by focusing on people who are not already Wikipedia contributors. This is because Wikipedia contributions currently follow a power law, with a relatively small number of prolific editors making most contributions [122, 158]. Prior work (e.g., [19, 91, 122]) and our interviews with veteran contributors suggest this is because people do not know they can contribute, are time constrained, are unfamiliar with Wikimarkup, feel unqualified, or feel their contributions are not important.

Overcoming these challenges and soliciting contributions from new people offers the potential to advance the community content creation feedback cycle in two ways. First, shifting the work of validating extractions onto newcomers frees experienced contributors to focus on other more demanding work. Second, it provides a quick and easy way for newcomers to make meaningful contributions. Bryant et al. report that newcomers become members of the Wikipedia community by participating in peripheral yet productive tasks that contribute to the overall goal of the community [19]. Making it easy for newcomers to examine and confirm Kylin extractions might therefore encourage more people to become active Wikipedia members.

This paper therefore focuses on soliciting contributions from people who have come to Wikipedia for some other reason, perhaps because they are seeking a specific piece of information or simply browsing out of curiosity, but did not already intend to work on Wikipedia. Contribution is therefore not a person's primary task. The challenge is then to design interfaces that make the ability to contribute by verifying Kylin extractions sufficiently visible that people choose to contribute, but not so obtrusive that people feel contribution is coerced (which would be seen as a violation of the Wikipedia community's goal of supporting free access to knowledge for everyone).

5.4 Interface Design and Refinement

In considering how to integrate the verification of Kylin extractions into Wikipedia articles, we note that Wikipedia already uses *cleanup tags* within articles [5]. Figure 5.3 shows an example

Figure 5.3: The Wikipedia community already uses *cleanup tags* to indicate opportunities for contribution, but these provide little assistance to potential contributors who are time constrained or unfamiliar with Wikimarkup.

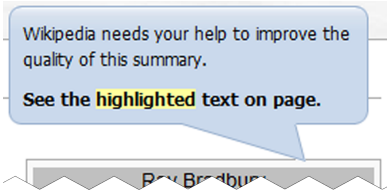


Figure 5.4: Each of our designs includes a callout drawing attention to the opportunity to help improve the article's infobox (see Figure 2 for the *icon* callout).

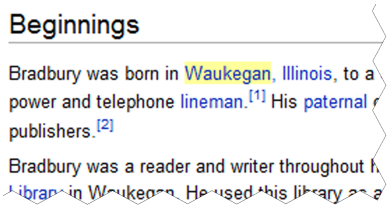


Figure 5.5: Our *highlight* design places a yellow highlight behind article text corresponding to potential extractions.

of one cleanup tag, drawing attention to the need to add references to an article. Although these tags illustrate the fact the Wikipedia community considers it appropriate to embed small indications of the need for work within articles, the current tags provide little or no assistance to potential contributors who are time-constrained or unfamiliar with Wikimarkup. In contrast, we aim to not only present the need for work within an article but also to leverage Kylin extractions so a person can contribute very quickly and easily (with just a few clicks). This section discusses general strategies we apply in all of our designs, as well as the details of our *popup*, *highlight*, and *icon* designs.

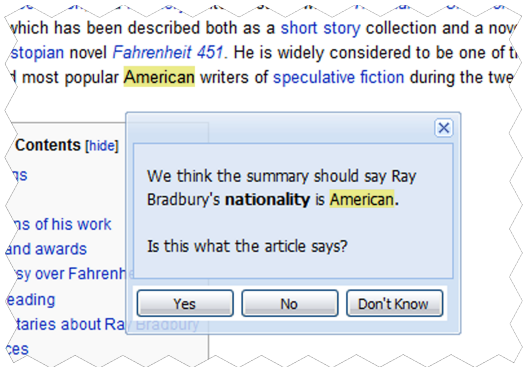


Figure 5.6: If interaction is initiated via the article, we annotate the extraction in the article and position our dialog to take advantage of the article context.

5.4.1 Ambiguity in a Non-Primary Task

Each of our designs explores a different approach to drawing attention to verifying Kylin extractions, but an important aspect of contribution as a non primary task is the fact many people will never notice the potential to contribute. All of our designs therefore never present unverified information in a way that might be mistakenly interpreted as a part of the article. Figure 5.2, for example, shows an infobox populated with the placeholder "Check our guess." Although prior work where ambiguity resolution is a part of the primary task might suggest other approaches (such as presenting an ordered list of potential values or the most likely value together with some indication of confidence) [40, 102, 140], it would be inappropriate to introduce the potential for a Wikipedia visitor to see a value in an infobox without realizing the value is unverified. All of our designs present unverified information within dialogs clearly separate from and floating above article content. The "Check our guess" placeholder is used throughout our designs whenever space should be allocated to content that is currently unverified.

5.4.2 Inviting Contributions from Visitors

All of our designs include a callout above the infobox explaining the opportunity to help improve the infobox (see figure 5.2 and figure 5.4). We originally used a banner across the top of the page, but early talk-aloud participants were unsure how their actions were improving the page. We switched

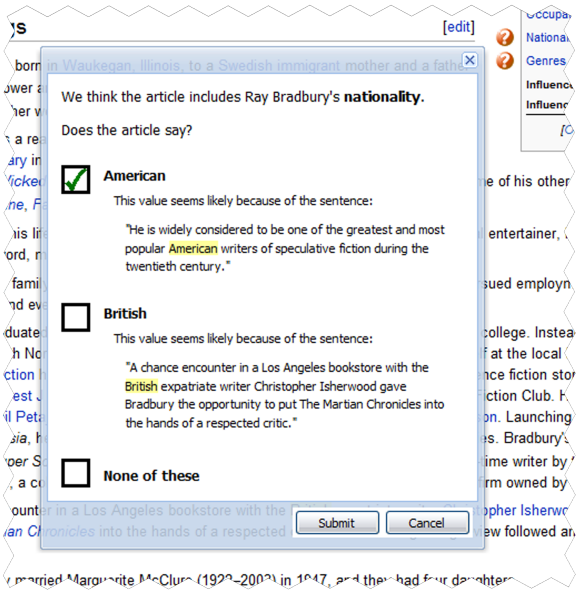


Figure 5.7: If interaction is initiated via the infobox, there may be different potential extractions at different locations. We therefore replicate the appropriate article context.

to the current callout to better draw attention to the infobox, consistent with the need to ensure people feel their contributions are important [19, 91, 122].

Popup Interface. Our first design is intended to solicit a greater number of contributions at the risk of being more obtrusive. It uses an *immediate* interruption coordination strategy [109], presenting a popup dialog as soon as a page is loaded. Dialogs are positioned adjacent to relevant content (as opposed to in the center of the browser). An immediate popup for each extraction in an article yielded an interface that was obviously too obtrusive. The first version tested in our informal talk-aloud sessions therefore randomly chose four non overlapping popups and presented those when the article was loaded. Talk-aloud participants still considered this overly obtrusive, and so our current design displays only one of the four popups at a time. If a person contributes via the popup, the next of the four is presented, but no additional popups are presented if a person closes a popup. The popups are non-modal, repositionable, do not scroll the browser or request focus, and otherwise do not interfere with article content except for the area obscured by the popup. Nevertheless, talk-aloud participants unanimously ranked this as the least acceptable interface.

Highlight Interface. Our second design is intended to better balance contribution rate against obtrusiveness. It uses a *negotiated* interruption coordination strategy [109], placing a yellow highlight behind text corresponding to potential extractions. Figure 5.5 illustrates this within the body of an article, and we also highlighted any "Check our guess" infobox placeholders. Mousing over either type of highlight presents a dialog allowing an indication of whether an extraction is correct. Responding to this dialog updates the infobox and removes any other obviated highlights.

Icon Interface. Our third design is intended to be the least obtrusive. It also uses *negotiated* interruption coordination [109], showing an icon for each potential extraction. These icons are placed on the left side of the infobox and along the left side of the article (as in figure 5.2). Upon mousing over an icon, the appropriate article text is highlighted and a dialog allows an indication of whether an extraction is correct. As in the highlight design, responding updates the infobox and removes any icons obviated by the response.

The biggest difference between *highlight* and *icon* pertains to intrusiveness. *Highlight* displays its cues in the article's body (highlighting words within the article) while *icon* does not disturb the contents of the article (displaying icons on the periphery). Three of our nine talk-aloud participants ranked *highlight* as their favorite, while six chose *icon*.

5.4.3 Presenting Ambiguity Resolution in Context

Our designs take two approaches to providing context for verifying Kylin extractions. The first, illustrated in figure 5.6, is presenting a dialog in the context of the article sentence from which Kylin obtained an extraction. We display the name of the infobox field in bold text and highlight the correspondence between the extracted value and the location of that value in the article. This dialog would look the same regardless of whether it was presented immediately as a part of the *popup* design, in response to mousing over the highlighted word "American" in the *highlight* design, or in response to mousing over an *icon* positioned off the left edge of figure 5.6. There is also an important subtlety in the wording of this dialog we revisit in discussing our Web search advertising deployments.

The second location our designs present extractions is in the infobox. It is important to take advantage of both locations to enhance the salience of opportunities to contribute, but the presen-

tation of context in the infobox is more difficult. For example, Kylin may have identified multiple sentences in an article that suggest potential values for a field, these sentences may not be located near the infobox, and they may also not be located near each other. Figure 5.7 shows our approach, duplicating a small amount of context within the ambiguity resolution dialog. The dialog highlights the extracted value in each sentence, and a confidence metric is used to indicate whether the match is "likely" or just "possible". In this case, the dialog is visible because a person moved their mouse over the *icon* in the upper-right corner of the figure. *Highlight* works similarly, but this dialog is not immediately presented in the *popup* interface. As a part of reducing the obtrusiveness of the *popup* design, we instead present a smaller dialog indicating that potential extractions are available. Clicking in that dialog then presents the larger verification dialog.

5.5 Web Search Advertising Deployment Study

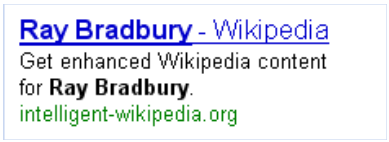


Figure 5.8: We used Web search advertising services to attract visitors to our pages. All of our visitors therefore had some other primary task, and we wanted to see whether they would spontaneously choose to contribute to Wikipedia.

Although our talk-aloud sessions were critical to refining our designs, it is difficult to imagine a laboratory study convincingly demonstrating whether our interfaces increase spontaneous contributions. We therefore developed a novel method using Web search advertising services to deploy our interfaces as an actual non primary task.

5.5.1 Procedure

We deployed a local Wikipedia mirror using a recent database dump and then randomly selected 2000 articles containing a *writer* infobox. We next used Kylin to extract values for the infobox fields. To ensure there would be an opportunity for contribution, we randomly removed up to ten

existing infobox fields from the set which Kylin had extracted. This is appropriate for evaluating our designs, as we are not yet making actual edits in Wikipedia.

We then used two Web search advertising services (Google AdWords and Yahoo Search Marketing) to place ads for each of the 2000 writers. Figure 5.8, for example, shows an advertisement that appeared in response to a Google query for "ray bradbury" while our ads were active. Clicking on our ads directed people to our local Wikipedia mirror, where we could add our interfaces. Note that our ads intentionally do not mention *contributing* to Wikipedia. We therefore believe that all of the people who visited our pages had some other primary task motivating their visit. We deployed four interfaces: our *popup*, *highlight*, and *icon* designs as well as a *baseline*. The *baseline* included a callout (analogous to figure 5.4) which prompted people to "Please edit this summary." Like Wikipedia's existing cleanup tags (see figure 5.3), *baseline* did not highlight text or otherwise ease contribution. Visitors were assigned to interface conditions in a round-robin manner.

Our proxy injected JavaScript for the appropriate interface into each page. We used AJAX calls to log a unique session identifier and time-stamped events (including the firing of page load and unload events, clicks on components of our interfaces, and interaction with the normal Wikipedia presentation of edit functionality in the *baseline* condition). We also injected a short questionnaire into each page. This appeared as a popup 60 seconds after the page load event. It asked participants whether they saw they could help improve the quality of the article, how disruptive they considered the prompts in the article, whether they would be willing to use the interface as an addition to Wikipedia, and then provided a field for freeform comments. We used referral information to remove from our analyses any visits that did not originate from our ads (including visits by our team and by automated crawlers).

5.5.2 Deployments

We initially deployed our study using Google AdWords, receiving 1131 visitors. Examining the freeform feedback from our survey revealed a potential misinterpretation of the wording used in our designs. Specifically, our initial dialogs said *"We think Ray Bradbury's nationality is American. Is this correct?"* Although we presented this in the context of the article and used highlighting to indicate the relationship to the article, we received comments like *"If I knew would I really need*

to look" and *"Please check with the Britannica"* that underscored visitor feelings they were unable to contribute. In retrospect it is clear our initial wording can be interpreted as asking for factual validation, and so we clarified the wording to *"We think the summary should say Ray Bradbury's nationality is American. Is this what the article says?"* We then conducted a small Google AdWords test of our revised wording with the *icon* design, acquiring another 285 visitors. Satisfied with the results of our change, we redeployed our study using Yahoo Search Marketing, receiving another 1057 visitors.

The next section quantitatively analyzes the results of our deployments to demonstrate the synergy of our pairing of community content creation with relation extraction, but we include this discussion to illustrate the challenges of designing for ambiguity resolution as a non-primary task. Further iteration could likely improve our designs, but we also believe significant future research is motivated by a need to better understand designing for non-primary tasks.

5.6 Demonstrating Synergistic Feedback

In order to show the synergy between community content creation and relation extraction, we analyze our results from two complementary perspectives: (1) we show we *accelerate community content creation* by using Kylin's relation extraction to significantly increase the likelihood that a person visiting a Wikipedia article as a part of some other primary task will spontaneously choose to help improve the article's infobox, and (2) we show we *accelerate relation extraction* by using contributions collected from people interacting with our designs to significantly improve Kylin's extraction performance.

5.6.1 Accelerating Community Content Creation

Table 5.1 summarizes the impact of our designs on the contribution rates of people who visited our pages as part of some other primary task. Recall *baseline* presented a callout asking visitors to help improve the quality of the infobox, analogous to Wikipedia's current cleanup tags, but did not leverage Kylin's relation extraction to ease contribution. We analyzed contribution likelihood using chi-squared tests in a sequential Bonferroni procedure, finding that all of our designs result in a significantly greater likelihood of contribution than *baseline* (*icon*: $\chi^2_{(1,N=1345)} = 23.0$,

$p < .001$, *highlight*: $\chi^2_{(1,N=1039)} = 53.0$, $p < .001$, *popup*: $\chi^2_{(1,N=1041)} = 55.4$, $p < .001$) and that *highlight* and *popup* yield a significantly greater likelihood of contribution than icon (highlight: $\chi^2_{(1,N=1432)} = 14.6$, $p < .001$, popup: $\chi^2_{(1,N=1434)} = 16.5$, $p < .001$). Analyzing the contributions per visit using Mann-Whitney tests in a sequential Bonferroni procedure finds the same differences (note that a large majority of people make no contributions, so finding the same differences is somewhat unsurprising).

We believe the lack of contribution in *baseline* is typical of people who come to Wikipedia for some reason other than a pre-existing intent to contribute, as Wikipedia's current cleanup tags provide little or no assistance to potential contributors who are time-constrained or unfamiliar with Wikimarkup. To further validate our analyses, we examined typical Wikipedia contribution rates. We analyzed three months of recent Wikipedia log data and found that only 1.6% of Wikipedia visits involve editing. Although it is impossible to know how many of these begin as a non-primary task, this contribution rate includes, for example, the work of people who dedicate extended periods of time to contribution as a primary task as well as the work of people using tools designed to help experienced and motivated contributors quickly make large numbers of edits [2]. Also relevant is the fact 32% of edits were anonymous, meaning 0.5% of Wikipedia visits involve anonymous editing. There are many potential reasons for anonymous editing, including the possibility a person is sufficiently new to the community that they do not have an account and the possibility a person had not intended to edit and so had not logged in to their account.

Our designs clearly succeed in *accelerating community content creation* by leveraging Kylin's relation extraction to obtain statistically significant improvements in contribution rates that herald practical implications. Every participant came to our pages with some other primary task, yet our *highlight* and *popup* designs, for example, yield an average of one contribution for every seven visits. Importantly, we obtained these results by emphasizing ease of contribution, not through coercion. Our results provide compelling initial evidence of the promise of using relation extraction to identify opportunities for people to quickly and easily contribute to community content creation systems.

Each of our designs promotes contribution in a different manner. Although our focus is that all of our designs were successful in leveraging Kylin's relation extraction to increase contribution, it is also interesting to consider differences in reactions to our designs. Figure 5.9 plots average intrusiveness (as reported by survey respondents) against the contribution likelihood for those designs.

	Baseline	Icon	Highlight	Popup
Visitors	476	869	563	565
Distinct Contributors	0	26	42	44
Contribution Likelihood	0%	3.0%	7.5%	7.8%
Number of Contributions	0	58	88	78
Contributions Per Visit	0	.07	.16	.14
Survey Responses	12	24	25	18
Saw I Could Help Improve	11/33 (33%)	30/73 (41%)	23/58 (40%)	24/52 (46%)
Intrusiveness (1:not — 5:very)	3.0	3.3	3.5	3.5
Willing to Use	11/33 (33%)	49/72 (68%)	34/57 (60%)	33/50 (66%)

Table 5.1: Summarizing the results of a total of 2473 visits to Wikipedia articles during our deployments. All of our designs significantly improve the likelihood of contribution.

We see the apparent correlation here as a motivation for future work further exploring this tradeoff, as we believe significant opportunities remain to explore designs that leverage relation extraction to solicit greater contribution rates without being perceived as intrusive. We also believe there are a number of questions to explore regarding how well different approaches will work with different types of data and in different communities.

5.6.2 Accelerating Information Extraction

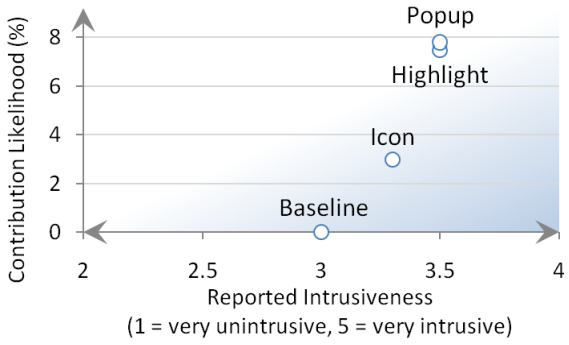


Figure 5.9: Contrasting contribution likelihood with reported obtrusiveness motivates future work exploring new designs that leverage relation extraction.

Having shown that relation extraction can amplify community content creation, we now demonstrate these contributions similarly improve relation extraction. We first examined the reliability of the 224 community created labels collected in our deployments. We removed 13 ambiguous labels, where our system had presented visitors with the entire sentence containing a correct value rather than the value itself. Of the remaining 135 extractions that visitors marked as correct, we found that 122 (90.4%) were indeed valid. Such high precision shows that making it easy for people to contribute does not necessarily mitigate quality. Of the 76 extractions that visitors indicated were incorrect, we found that only 44 (57.9%) were actually errors. This high false negative rate likely indicates people were conservative during validation, but may also be due in part to confusion over factual verification versus extraction validation (as discussed in the previous section).

We next examined whether these noisy community-created labels actually improve Kylin's relation extraction performance. Because Kylin learns by analyzing existing infoboxes, we expected the impact of community-created labels to diminish if there were numerous existing infobox examples available. Thus, we test the effect of the community labels with models trained on 5, 10, 25, 50, and 100 existing infobox examples. We chose these numbers because 72% of infobox classes have 100 or fewer articles and 40% have 10 or fewer articles. Furthermore, Kylin frequently cannot obtain a training example for every field from an article containing an infobox. For example, an infobox may not contain a value for a field or there may not be a sentence in the article that matches the field. When given an article with a *writer* infobox, for example, Kylin was able to generate a positive training example for only an average of 14.5% of the attributes.

The performance of any individual extractor is difficult to interpret, and the performance of extractors for different infobox fields cannot be directly compared. The nature of sentences containing a birthdate, for example, may make that date more (or less) difficult to extract than a person's nationality. We conducted our experiment with the five *writer* fields for which we obtained the greatest number of positive examples during our Web advertising deployment. We trained Kylin using a set of randomly-selected existing infobox examples for each field and tested the resulting extractor against 200 articles, the fields in which we had manually labeled. We then added the community created labels (both correct and incorrect) and repeated the test. To minimize errors caused by sampling, we repeated this process for ten trials with different initial infobox examples. As an outcome measure, we chose the *area under the precision-recall curve*, a common summary statistic

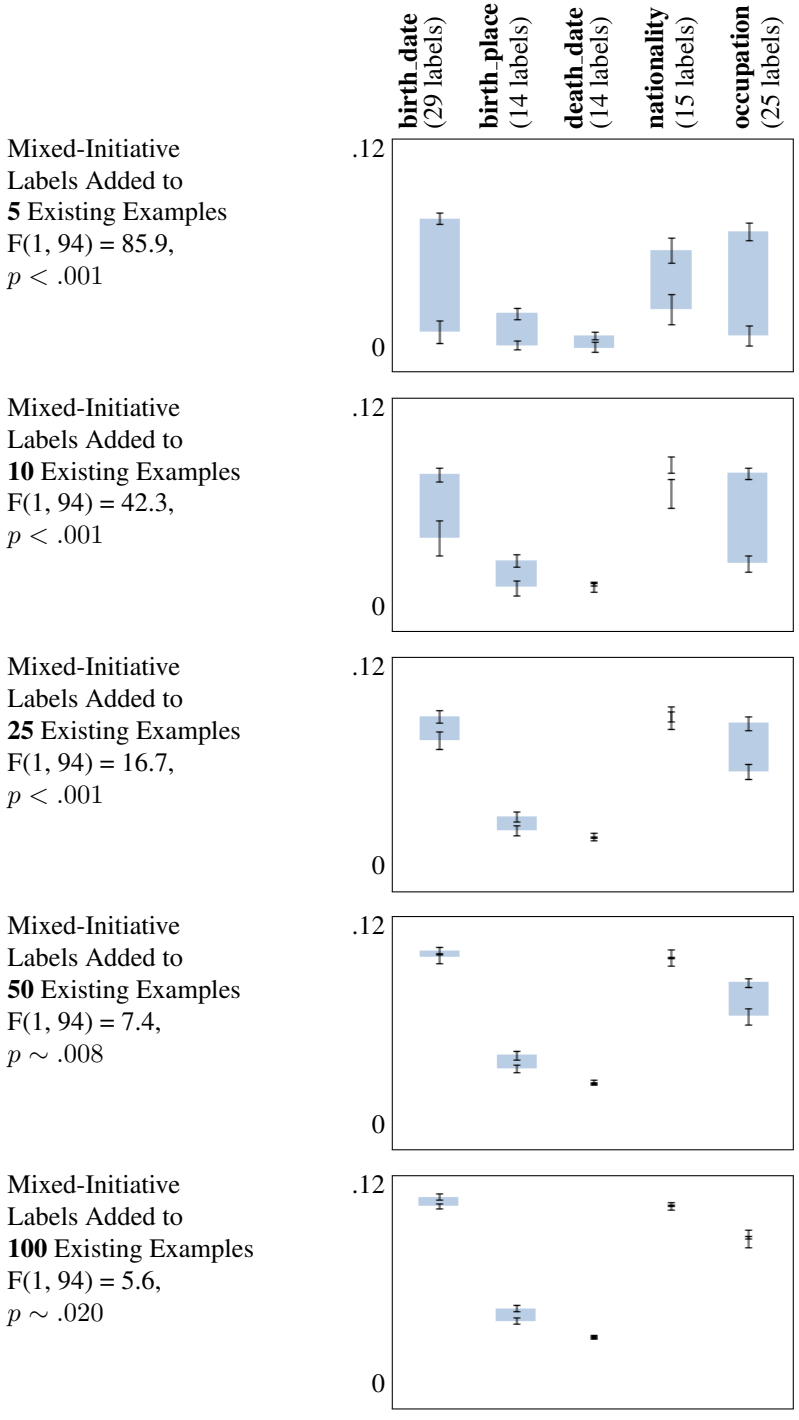


Table 5.2: Adding our community-created labels to examples found from existing article infoboxes significantly improves Kylin's extraction performance in all five groups of trials (as measured by the area under the precision-recall curve). This impact is most dramatic when relatively few existing infoboxes are available. This is the typical case in Wikipedia, where 72% of inbox classes appear in less than 100 articles.

for relation extraction performance.

Table 5.2 summarizes our results. We show standard error bars around the mean area under the precision recall curves from the ten trials. The means are connected by a wide blue bar whenever a paired *t* test indicated a statistically significant improvement from the addition of the community created labels. Because we have noted that different fields can vary in the difficulty of their extraction, our analysis focuses on the impact of adding community-created labels to Kylin extractors trained on differing numbers of existing infobox examples. We analyze each group of trials using a mixed-model analysis of variance, treating field as a random effect. As reported in table 5.2, our analyses show that the addition of community created labels significantly improves the area under the precision-recall curve in all five groups of trials.

These results provide strong initial evidence of the second half of our synergy, that despite containing errors, community-created labels *accelerate relation extraction* by significantly improving Kylin's extraction performance. Our community-created labels most dramatically improved extraction performance when relatively few existing infobox examples were available, and we have noted that, indeed, most Wikipedia infobox classes have relatively few existing examples available.

5.7 Related Work

Prior work has explored why people contribute to Wikipedia and what the implications of those motivations are for the Wikipedia culture [19, 91, 122]. Throughout this paper we have discussed designing for a culture that is motivated by altruism, supporting free access to knowledge for everyone, reputation, and indirect author recognition. Work by Cosley et al. [37] has examined the problem of finding appropriate articles for experienced contributors to work on, based in the idea that a person's editing history provides insight into what other articles they might be interested in editing. Instead of targeting experienced Wikipedia contributors, we leverage relation extraction to solicit contributions from people who otherwise would be unlikely to contribute. Our approach frees experienced contributors to focus on more challenging work and provides a path for newcomers to become active members, consistent with Bryant et al.'s finding that newcomers become members of the Wikipedia community by participating in peripheral yet productive tasks that contribute to the overall goal of the community [19].

DeRose et al. [45] and Doan et al. [106] each propose different approaches to building communities based on both human and automated contributions. DeRose et al. build their MadWiki system with structured slots, which are attribute/value pairs expressed in terms of paths over entity relationship representations of database schemas. Doan et al. focus on schema matching, examining several design dimensions and proposing that one approach to obtaining contributions might be to require some amount of work before providing a service. In addition to examining a different type of inference, we take a different perspective by placing the needs and norms of the existing Wikipedia community on equal footing with the needs of relation extraction systems. We work with Wikipedia's existing content format and community norms because we believe the full benefits of pairing relation extraction with community content creation can be realized only by reinforcing both feedback cycles.

Von Ahn and Dabbish's *games with a purpose* channel player entertainment into productive work, such as labeling images through a game in which players guess what words other players will guess in response to an image [156, 157]. Although the verification of relation extraction does not easily fit into the game templates described by von Ahn and Dabbish [157], we believe the more interesting contrast is a difference in perspective regarding community contribution. Von Ahn and Dabbish's games are ultimately a deception, disguising work as a game. In contrast we highlight the opportunity to contribute meaningful work to a community, leveraging our synergy with relation extraction to make contribution easy. The approaches are clearly complementary, but we believe it is notable that our approach provides a path for newcomers to become active members of a community, at which point they may choose to take on more challenging work (work that cannot be reduced to a game or to a handful of clicks).

Prior work has explored interfaces for ambiguity resolution, including Mankoff et al.'s OOPS [102], Shilman et al.'s CueTIP [140], and Culotta et al.'s examination of corrective feedback in information extraction [40]. Such work has focused on ambiguity resolution as a part of the primary task, at least in the sense that the primary task cannot continue until the ambiguity is resolved. Although we leverage many of the techniques developed in such work, we have also shown that designing for ambiguity resolution as a non-primary task introduces new challenges.

5.8 Conclusion

To improve relation extractors experts often need to invest a prohibitive amount of effort. This chapter presents an alternative approach which leverages an online community of non-experts. In particular, it proposes a novel synergistic method for jointly amplifying community content creation and learning based relation extraction. By enabling both techniques to exploit the same edits, two interlocking feedback cycles accelerate each other. We have demonstrated this synergy with two complementary analyses: (1) we show we *accelerate community content creation* by using Kylin's relation extraction to significantly increase the likelihood of contribution by people visiting Wikipedia while engaged in some other primary task, and (2) we show we *accelerate relation extraction* by using community-created contributions as training examples to significantly improve Kylin's extraction performance. Taken together, these analyses provide initial but compelling evidence of our proposed synergy.

Chapter 6

DISCUSSION

We seek to create a tool that would dramatically reduce the human effort necessary to create a relation extractor. More specifically, the tool would make it possible to create a relation extractor in less than one hour, using no previously annotated data. To achieve this goal, we proposed a set of rich user interactions, some based on more accurate approaches to weak supervision, and demonstrated their effectiveness. In this chapter, we reflect on the lessons learned, compare the different techniques to each other, and discuss their role in our envisioned tool.

Our first interaction, providing databases of relation instances, tries to reduce human effort most dramatically, but requires learning with weak supervision. Since learning with weak supervision is challenging and most existing approaches are inaccurate, we proposed two new approaches, MULTIR and LUCHS and demonstrated their effectiveness in experiments. In many cases, our approaches enable us to create extractors with virtually no effort. They easily scale to thousands of relations, and they can successfully handle noise in the data.

In particular, MULTIR shows that we can learn an accurate sentence-level extractor that uses aggregate-level reasoning to model weak supervision during learning. This is quite significant, since previous approaches require aggregate-level reasoning for both learning and inference, making extraction much less efficient. More importantly, our experiments confirm that our approach is much more accurate, especially on individual sentences, and unlike previous ones can handle overlapping relations. Our experiments, however, also give us insights that point towards further opportunities for improvement. MULTIR, like previous approaches, uses multi-instance learning to combat noise, and thus assumes that at least one of the sentences containing an entity pair from the database expresses the relation in the database. Manual inspection, however, shows that this assumption is often violated, causing many false parameter updates during learning. Moreover, the model currently does not take into account the frequency of sentences with entity pairs, giving too much importance to sentences with entity pairs that occur only once. For example, consider a sen-

tence containing an entity pair that exists as a relation in our database. If the entity pair is contained in no other sentence, then the algorithm will always perform a parameter update if it does not already predict the relation for that sentence. On the other hand, if the same entity pair is contained in many other sentences then the chance is high that the relation is already predicted from another sentence, in which case it will not perform a parameter update. As a consequence, MULTIR tends to be conservative, often preferring not to extract, and one must adjust its sensitivity by varying the amount of randomly selected negative data. In chapter 7, we will discuss how we hope to solve this problem in future work.

While MULTIR handles noise through multi-instance learning, another approach is to simply reduce noise by being more selective about which text corpus and database to align. In our work on LUCHS, we treat Wikipedia infoboxes as databases and align them to the text of their corresponding articles. Since a Wikipedia article and its infobox often state the same facts, there are few incorrect matches. The number of incorrect matches is even reduced further by considering only the first ten sentences of each article. Although such filtering dramatically reduces noise in our training data, it also introduces a new challenge, sparseness. For most relations, there remain few heuristically matched training sentences. LUCHS handles such sparseness by automatically inducing relation-specific word lexicons from a large corpus of Web lists and then adding these as features to bias an extractor. Our experiments show that, combined with our proposed technique of cross-training, this approach substantially improves the quality of thousands of relation extractors. However, there are also weaknesses. First, we note that the system currently only works for Wikipedia pages. For example, LUCHS assumes that each page corresponds to exactly one schema and that the subject of relations on a page are the same. Also, LUCHS makes predictions on a token basis, thus sometimes failing to recognize larger segments. To remove these limitations we plan to enhance other extractors such as MULTIR with LUCHS' induced lexicons.

More generally, however, we see two important drawbacks of our weakly supervised approaches MULTIR and LUCHS. First, both suffer from their inability to recover deeper linguistic structure. For example, consider the following sentences.

1. John was killed by Tim.

2. John was murdered by Tim.

3. John was killed by a group that included Tim.

4. John was murdered by a group that included Tim.

MULTIR's features, originally defined by Mintz et al. [112], render each of these sentences very differently, hindering generalization. Even if MULTIR sees sentences like 1, 2, and 3 during training, it will be incapable of extracting 4. In other words, MULTIR is unable to recognize that the sentences can be decomposed into different parts that can then be recombined in different ways. Second, neither approach leverages existing components and linguistic knowledge, which could help avoid many errors. While both use a syntactic parser, none uses a coreference resolver. Furthermore, none uses linguistic resources like WordNet or a set of general syntactico-semantic rules. We believe that the best way to overcome these drawbacks, is to make it easy for users to directly teach our system the necessary linguistic knowledge.

When there exists no database that can be used for weak supervision or when the quality of a weakly-supervised extractor is low, we propose to enable users to directly write extraction rules. Our system, INSTAREAD allows users to write rules in logic and then analyze their behavior on large datasets. While that requires more work from the user than merely providing a database, rules are a very direct way of expressing prior knowledge and humans tend to be extremely good at recognizing structure. In many cases, a few simple rules suffice to create an extractor of acceptable quality. Since finding good rules can still be challenging, INSTAREAD includes a set of features to make that quick and easy. For example, INSTAREAD is able to evaluate rules on large datasets instantly, by using database techniques such as indexing and dynamic query optimization; INSTAREAD's language is also very expressive with support for operators such as $\neg$, $\wedge$, $\vee$, $\forall$, and $\exists$, and predicates to use the output of different system components. Our experiments, however, showed that that alone is not enough. Two additional features turned out to be very important: First, INSTAREAD makes it possible to execute one iteration of bootstrapping in order to discover new rules. Second, INSTAREAD makes it possible to decompose rules by defining new predicates and then re-using these predicates in other rules. Not only does this make it possible to represent a set of rules more compactly, it is also necessary for scaling rule-based extraction. In our experiments, INSTAREAD was very effective, even for relations where our weakly supervised extractors failed.

Nonetheless, INSTAREAD's approach does have shortcomings. Foremost, the system is currently not able to adequately deal with ambiguities. While our learning-based systems handle ambiguities by weighting different features, INSTAREAD's rules are deterministic. In our experiments on four binary relations, such ambiguities did not arise, but for many relations they are common. For example, a unary extractor for cities needs to trade off different evidence to determine if the token 'Paris' occurring in a sentence refers to a city or a person. While it may be possible to add weights to INSTAREAD's rules, it would be extremely difficult for a human to set such weights manually. One future goal is therefore to integrate machine learning into the system, so that weights can be set automatically. Moreover, we note that creating extractors with INSTAREAD can still be time-intensive. For many applications, even one hour of development time per relation may be too long. We see, however, several ways to further accelerate the process. For example, there is currently no support for assisting users in decomposing rules, and we believe that clustering algorithms could help to simplify this task. Furthermore, we noticed that it becomes increasingly difficult to manage rules as their number grows. At some point, a user is no longer aware which rules she has already written and reading a set of existing rules is difficult and slow. In sich situations, it would be great if the system automatically suggested actions such as to merge rules, to eliminate duplicate rules, or to extract higher-level rules. Finally, INSTAREAD would become more useful if rules could be easily shared between users. Many rules are not relation-specific and many users might be interested in extracting the same relations.

Even with the suggested improvements, however, rule writing will require expert skill. In our final work on SMARTWIKI we therefore looked at how we can leverage non-expert users to improve an extractor. A central challenge is that the interactions had to be extremely simple, so that many users would be willing to perform them. Although both INSTAREAD and SMARTWIKI attempt to optimize the interaction between human and system, their foci are different: INSTAREAD tries to maximize the diversity of changes a user can make to an extractor with minimal effort, while SMARTWIKI tries to maximize the number of users giving feedback. Our experiments with random visitors to Wikipedia show that it is possible to design interfaces that dramatically increase the likelihood that a person spontaneously chooses to contribute. Furthermore, they show that the feedback collected that way can significantly improve extractors.

Our use of Web search advertising services was a powerful way to expose people to our inter-

faces as a non primary task, but it is clear that we need to address the more complete integration of SMARTWIKI into a variety of sites in future work. Motivations for contribution may vary among different Web communities, but our methods suggest that our approach generalizes beyond Wikipedia. Our contributions were not solicited from people in the Wikipedia community (which may be a somewhat atypical Web community), but were instead solicited from people using the Web in their everyday primary tasks (due to our use of Web search advertising services). Prior research on contribution to different types of community content sites also reveals many similarities: contribution is often highly skewed, and so successful sites need to provide value to visitors, make the need for contribution visible, ensure it is easy to contribute (especially for newcomers), and ensure people perceive the contributions as meaningful, all principles that our synergistic approach is designed to address.

So, in summary, which technique is most effective? We have discussed advantages and weaknesses of each approach, and it is difficult to say which one is 'best'. That depends on the relation to be extracted and the resources available. Also, the techniques are not mutually exclusive; in fact, they can be combined in various ways to further improve performance. Nonetheless, we have general recommendations as to how one can proceed in applying them. MULTIR requires virtually no user effort, and, as our experiments in chapter 2 demonstrated, can learn high-precision extractors for a variety of relations. If precision is more important than recall and there exists an appropriate database of relation instances, we therefore suggest to first try MULTIR. In the case where one needs to learn a larger number of relation extractors and does not have appropriate entity type recognizers for some relation arguments, adding the LUCHS technique to a weakly supervised learner usually yields an improvement in both precision and recall of several percentage points. The major problems with our weakly supervised techniques is that there often does not exist an appropriate database of relation instances, and that recall is often only in the $10 - 50\%$ range. When these problems are critical, we suggest to apply INSTAREAD. INSTAREAD requires more user effort than the weakly supervised techniques, but it makes it possible to create high-quality extractors even when other techniques fail. Also, the trade-off between extraction quality and user effort can easily be controlled by the user. Our experiments show that 55 minutes of development time suffice to create extractors that greatly outperform ones learned by MULTIR. On a particularly difficult relation for which there was no database of instances available, INSTAREAD reached 41.2% recall at 97.3% pre-

cision. Finally, in the case where one has access to a large online community, it may be possible to use SMARTWIKI to collect validative feedback for improving a learned extractor. Our experiments show that one can obtain a high contribution rate and that more than 90% of extractions labeled correct were indeed correct. When applicable, SMARTWIKI therefore provides the opportunity to easily enhance precision.

Chapter 7

FUTURE WORK

The discussion of our systems in the previous chapter already mentioned several ways in which they could be improved. In this chapter, we would like to take a step back and look at our challenges more broadly. We originally set ourselves the goal of creating an extractor in one hour. What should be our next milestone? Is it possible to enables users to create better extractors in only 5 minutes? What fundamental change needs to happen so that we can reach that milestone?

We believe that our general approach of enabling richer interactions with more accurate machine learning has great potential beyond the techniques presented in this dissertation. With a few enhancements, it is quite possible that one can create quality extractors in just a few minutes. We believe that two such enhancements are critical: First, we need a deep integration of different techniques. Second, we must continue to make our models for learning more accurate.

The most critical enhancement necessary is to more deeply integrate the different techniques. One example is an integration of learning, as provided by MULTIR, and interactive rule writing, as provided by INSTAREAD. Users of MULTIR may have prior assumptions about how a database could be more accurately aligned with a text corpus, and could express these assumptions through rules. Furthermore, a weakly supervised learning algorithm could leverage existing rules. Such existing rules could make an extractor more accurate and help reduce the search space. Not only can rules help learning, however, but learning can also help create rules. Recall that our deterministic rules made it difficult to deal with ambiguities. Using learning, it is possible to learn weights for existing rules to more effectively handle ambiguities. With appropriate regularization, it may also be possible to learn sparse sets of rules that could be automatically presented to a user for validation. Note that such combinations of learning and rule writing are not only relevant to situations where we can apply weak supervision from a database. Clustering techniques, too, can be combined with rule writing, to reduce human effort when no database is available. In particular, a clustering technique could speed up INSTAREAD's rule decomposition, which is currently entirely manual.

Another example of deep integration would be to enhance INSTAREAD's bootstrapping operation by adding more linguistic knowledge in the form of syntactic rules, word classes, or even entire components for coreference resolution, entity linking, or other natural language processing tasks. Such an integration would likely not only increase precision and recall, but also change the nature of the rules recommended by the tool. Since the integrated components would already capture many kinds of syntactic variations, there would be fewer system suggestions and they might be more general (for example paraphrases instead of syntactic variations), thus further reducing human effort. Such deep integration of linguistic knowledge could of course also be useful for weakly supervised or unsupervised learning-based techniques.

A deep integration of different components may also give us an opportunity to increase the quality of each component through joint-inference. Our experiments in Section 4.7.4 show the promise of a joint inference approach to relation extraction and syntactic parsing. In the future, we would like to extend that to other components such as named-entity classification and linking, and coreference resolution.

In the long run, our deep integration should also encompass ontology creation. We note that our extractor's performance may benefit substantially from an ontology. Furthermore, as we showed in Section 3.5.6, our current extractors can also greatly facilitate learning an ontology. We therefore plan to deeply integrate extractor learning and ontology inference, hence jointly learning ontology and extractors. We further believe that our general approach of providing richer interactions with more accurate learning is also applicable to ontology creation, and may enable us to create higher quality ontologies with limited human effort.

Besides such integrations of different components, we also hope to make our models of weak supervision even more accurate. MULTIR and previous systems [127, 20] assume that for each relation instance in a knowledge-base, at least one of the matching sentences expresses that relation. Our inspection of weakly supervised training data showed, however, that this assumption is frequently violated, and that in fact, for a large number of relation instances in a knowledge base none of the matching sentences express the relation. A more accurate assumption may be to assume that with a certain percentage chance a sentence matching a relation instance expresses that relation. The percentage would be relation-specific, and the model could then be trained under the constraint that these percentages are roughly satisfied. We believe that this change could reduce the number

of incorrect updates during learning; it could better leverage entity pairs that occur frequently while reducing the weight of entity pairs that occur only once; and it could make our model less sensitive to noise in the data, thus making weak supervision more accurate.

Chapter 8

CONCLUSIONS

Information extraction technology promises exciting possibilities, such as question-answering systems that combine facts stated on different pages, analytical systems that give us new insights into society and human behavior, and systems that aggregate experimental results from research publications to accelerate progress in drug discovery. Today, however, there does not exist a system that can reliably convert text into a knowledge-base, and the task is challenging. A key problem is relation extraction. While most successful approaches use supervised machine learning from labeled training data, the necessary user labeling effort is prohibitive for constructing Web-scale knowledge bases.

This dissertation makes progress towards solving this dilemma by considering a range of rich human interactions with an extraction system. One key interaction is providing databases of relation instances. Weak supervision from such databases may enable creating an extractor with minimal human effort. Unfortunately, existing methods tend to suffer from low precision and recall. When weak supervision fails, experts can write extraction rules manually. Without adequate interactions to facilitate this process, however, it can be a tedious endeavor. To avoid the need for expensive expert input, we would also like to leverage large online communities to improve an extractor. Which interactions motivate them to contribute?

Our work provides novel solutions to all these challenges, and shows how a tighter integration of different forms of human feedback and techniques based on machine learning, is bringing us closer towards scalable systems which reliably convert natural language text into knowledge bases.

BIBLIOGRAPHY

[1] Mediawiki. http://www.mediawiki.org.

[2] Wikipedia: Autowikibrowser. http://en.wikipedia.org/wiki/Wikipedia:AutoWikiBrowser.

[3] Wikipedia: Be bold. http://en.wikipedia.org/wiki/Wikipedia:Be_Bold.

[4] Wikipedia: Bots. http://en.wikipedia.org/wiki/Wikipedia:Bots.

[5] Wikipedia: Cleanup tags. http://en.wikipedia.org/wiki/Wikipedia:Template_messages/Cleanup.

[6] Eugene Agichtein and Venkatesh Ganti. Mining reference tables for automatic text segmentation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 20–29, 2004.

[7] Eugene Agichtein and Luis Gravano. *Snowball*: extracting relations from large plain-text collections. In *Proceedings of the ACM Conference on Digital Libraries (DL)*, pages 85–94, 2000.

[8] Douglas E. Appelt and Boyan Onyshkevych. The common pattern specification language. In *Proceedings of the TIPSTER workshop*, 1998.

[9] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of the International Semantic Web Conference (ISWC) and Asian Semantic Web Conference (ASWC)*, pages 722–735, 2007.

[10] Shumeet Baluja, Rohan Seth, D. Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceedings of the Conference on World Wide Web (WWW)*, pages 895–904, 2008.

[11] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2670–2676, 2007.

[12] Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 28–36, 2008.

[13] Kedar Bellare, Gregory Druck, and Andrew McCallum. Alternating projections for learning with expectation constraints. *CoRR*, abs/1205.2660, 2012.

[14] Kedar Bellare and Andrew McCallum. Learning extractors from unlabeled text using relevant databases. In *Proceedings of the Sixth International Workshop on Information Integration on the Web (IIWeb)*, 2007.

[15] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[16] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. Label propagation and quadratic criterion. In Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors, *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006.

[17] Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks*, 30(1-7):107–117, 1998.

[18] Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jennifer C. Lai, and Robert L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[19] Susan L. Bryant, Andrea Forte, and Amy Bruckman. Becoming wikipedian: transformation of participation in a collaborative online encyclopedia. In *Proceedings of the ACM Conference on Supporting Group Work (GROUP)*, pages 1–10, 2005.

[20] Razvan Bunescu and Raymond Mooney. Learning to extract relations from the web using minimal supervision. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2007.

[21] Razvan C. Bunescu and Raymond J. Mooney. Subsequence kernels for relation extraction. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2005.

[22] Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: exploring the power of tables on the web. *Proceedings of the International Conference on Very Large Databases (VLDB)*, 1(1):538–549, 2008.

[23] Andrew Carlson, Justin Betteridge, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupling semi-supervised learning of categories and relations. In *Proceedings of the NAACL HLT Workskop on Semi-supervised Learning for Natural Language Processing*, 2009.

[24] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[25] Andrew Carlson, Scott Gaffney, and Flavian Vasile. Learning a named entity tagger from gazetteers with the partial perceptron. In *Proceedings of the AAAI Spring Symposium on Learning by Reading and Learning to Read*, 2009.

[26] John M. Carroll and Mary Beth Rosson. Paradox of the active user. In John M. Carroll, editor, *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, chapter 5, pages 80–111. Bradford Books/MIT Press, 1987.

[27] Ming-Wei Chang, Lev-Arie Ratinov, and Dan Roth. Guiding semi-supervision with constraint-driven learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2007.

[28] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

[29] Hao Chen and Susan T. Dumais. Bringing Order to the Web: Automatically Categorizing Search Results. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, pages 145–152, 2000.

[30] Harr Chen, Edward Benson, Tahira Naseem, and Regina Barzilay. In-domain relation discovery with meta-constraints via posterior regularization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 530–540, 2011.

[31] Jinxiu Chen, Dong-Hong Ji, Chew Lim Tan, and Zheng-Yu Niu. Unsupervised relation disambiguation using spectral clustering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.

[32] Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, and Shivakumar Vaithyanathan. Systemt: An algebraic approach to declarative information extraction. In *Proceedings of the Annual Meetings of the Association for Computational Linguistics (ACL)*, pages 128–137, 2010.

[33] Peter Clark and Phil Harrison. Recognizing textual entailment with logical inference. In *Proceedings of the Text Analysis Conference (TAC)*, 2008.

[34] William W. Cohen and Sunita Sarawagi. Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 89–98, 2004.

[35] 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 (EMNLP)*, 2002.

[36] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the International Conference for Machine Learning (ICML)*, pages 160–167, 2008.

[37] Dan Cosley, Dan Frankowski, Loren G. Terveen, and John Riedl. Suggestbot: using intelligent task routing to help people find work in wikipedia. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, pages 32–41, 2007.

[38] Mark Craven and Johan Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 77–86, 1999.

[39] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–716, 2007.

[40] Aron Culotta, Trausti T. Kristjansson, Andrew McCallum, and Paul A. Viola. Corrective feedback and persistent learning for information extraction. *Artificial Intelligence*, 170(14-15):1101–1122, 2006.

[41] Hamish Cunningham. Gate, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254, 2002.

[42] Edward Cutrell and Zhiwei Guan. What are you looking for? an eye-tracking study of information usage in Web search. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, 2007.

[43] Edward Cutrell, Daniel C. Robbins, Susan T. Dumais, and Raman Sarin. Fast, Flexible Filtering with Phlat. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, pages 261–270, 2006.

[44] Dmitry Davidov, Ari Rappoport, and Moshe Koppel. Fully unsupervised discovery of concept-specific relationships by web mining. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, 2007.

[45] Pedro DeRose, Xiaoyong Chai, Byron J. Gao, Warren Shen, AnHai Doan, Philip Bohannon, and Xiaojin Zhu. Building community wikipedias: A machine-human partnership approach. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 646–655, 2008.

[46] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[47] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. Ace program - task definitions and performance measures. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 837–840, 2004.

[48] Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

[49] Xin Dong, Alon Y. Halevy, and Jayant Madhavan. Reference Reconciliation in Complex Information Spaces. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 85–96, 2005.

[50] Mira Dontcheva, Steven M. Drucker, Geraldine Wade, David Salesin, and Michael F. Cohen. Summarizing Personal Web Browsing Sessions. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 115–124, 2006.

[51] Gregory Druck, Burr Settles, and Andrew McCallum. Active learning by labeling features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 81–90, 2009.

[52] Susan T. Dumais, Edward Cutrell, Jonathan J. Cadiz, Gavin Jancke, Raman Sarin, and Daniel C. Robbins. Stuff I've Seen: A System for Personal Information Retrieval and Re-Use. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 72–79, 2003.

[53] Susan T. Dumais, Edward Cutrell, and Hao Chen. Optimizing Search by Showing Results In Context. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, pages 277–284, 2001.

[54] Benjamin Van Durme and Marius Pasca. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1243–1248, 2008.

[55] Nadav Eiron and Kevin S. McCurley. Analysis of Anchor Text for Web Search. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 459–460, 2003.

[56] Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Methods for domain-independent information extraction from the web: An experimental comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 391–398, 2004.

[57] Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.

[58] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545, 2011.

[59] David A. Ferrucci and Adam Lally. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

[60] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370, 2005.

[61] Paul Fodor, Adam Lally, and David A. Ferrucci. The prolog interface to the unstructured information management architecture. *CoRR*, abs/0809.0680, 2008.

[62] Marjorie Freedman, Lance A. Ramshaw, Elizabeth Boschee, Ryan Gabbard, Gary Kratkiewicz, Nicolas Ward, and Ralph M. Weischedel. Extreme extraction - machine reading in a week. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1437–1446, 2011.

[63] Dayne Freitag. Toward general-purpose learning for information extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 404–408, 1998.

[64] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[65] Johannes Fürnkranz. Exploiting structural information for text classification on the www. In *Proceedings of the International Symposium on Intelligent Data Analysis (IDA)*, pages 487–498, 1999.

[66] Ryan Gabbard, Marjorie Freedman, and Ralph M. Weischedel. Coreference for learning to extract relations: Yes virginia, coreference matters. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, pages 288–293, 2011.

[67] Krzysztof Gajos, David B. Christianson, Raphael Hoffmann, Tal Shaked, Kiera Henning, Jing Jing Long, and Daniel S. Weld. Fast and robust interface generation for ubiquitous applications. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing (Ubicomp)*, pages 37–55, 2005.

[68] Krzysztof Gajos and Daniel S. Weld. Supple: automatically generating user interfaces. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, pages 93–100, 2004.

[69] Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049, 2010.

[70] Zoubin Ghahramani and Katherine A. Heller. Bayesian sets. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2005.

[71] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

[72] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the ACM Conference on Digital Libraries (DL)*, pages 89–98, 1998.

[73] Jonathan Grudin. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM*, 37(1):92–105, 1994.

[74] Zhiwei Guan and Edward Cutrell. An Eye Tracking Study on How People Search When the Target is Not Shown on Top of the List. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, 2007.

[75] Joseph Hassell, Boanerges Aleman-Meza, and Ismailcem Budak Arpinar. Ontology-driven automatic entity disambiguation in unstructured text. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 44–57, 2006.

[76] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Conference on on Computational Linguistics (COLING)*, pages 539–545, 1992.

[77] Lynette Hirschman, Alexander A. Morgan, and Alexander S. Yeh. Rutabaga by any other name: extracting biological names. *Journal of Biomedical Informatics*, 35(4):247–259, 2002.

[78] Raphael Hoffmann, Saleema Amershi, Kayur Patel, James Fogarty, and Daniel S. Weld. Amplifying community content creation using mixed-initiative information extraction. In *Proceedings of the International Conference on Human Factors in Computing Systems (CHI)*, 2009.

[79] Raphael Hoffmann, James Fogarty, and Daniel S. Weld. Assieme: Finding and leveraging implicit references in a web search interface for programmers. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, 2007.

[80] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 541–550, 2011.

[81] Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. Learning 5000 relational extractors. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 286–295, 2010.

[82] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, pages 159–166, 1999.

[83] Jeff Huang, Oren Etzioni, Luke Zettlemoyer, Kevin Clark, and Christian Lee. Revminer: An extractive interface for navigating reviews on a smartphone. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, 2012.

[84] David F. Huynh, Robert C. Miller, and David R. Karger. Enabling web browsers to augment web sites' filtering and sorting functionalities. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 125–134, 2006.

[85] Heng Ji, Ralph Grishman, and Hoa Trang Dang. An overview of the tac2011 knowledge base population track. In *Proceedings of the Text Analysis Conference (TAC)*, 2011.

[86] Jing Jiang and ChengXiang Zhai. A systematic exploration of the feature space for relation extraction. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computation Linguistics (HLT-NAACL)*, pages 113–120, 2007.

[87] Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 668–677, 1998.

[88] Stanley Kok and Pedro Domingos. Extracting semantic networks from text via relational clustering. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 624–639, 2008.

[89] Jayant Krishnamurthy and Tom M. Mitchell. Weakly supervised training of semantic parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2012.

[90] Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, Shivakumar Vaithyanathan, and Huaiyu Zhu. Systemt: a system for declarative information extraction. *SIGMOD Record*, 37(4):7–13, 2008.

[91] Stacey Kuznetsov. Motivations of contributors to wikipedia. *ACM Computers and Society*, 36(2):1–7, 2006.

[92] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

[93] Ni Lao, Tom M. Mitchell, and William W. Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirial Methods in Natural Language Processing (EMNLP)*, pages 529–539, 2011.

[94] Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W. Cohen. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the Conference on Empirial Methods in Natural Language Processing (EMNLP)*, 2012.

[95] Claudia Leacock, Martin Chodorow, and George A. Miller. Using corpus statistics and wordnet relations for sense identification. *Computational Linguistics*, 24(1):147–165, 1998.

[96] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, CONLL Shared Task '11, pages 28–34, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[97] Xiao Li, Ye-Yi Wang, and Alex Acero. Learning query intent from regularized click graphs. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 339–346, 2008.

[98] Percy Liang, A. Bouchard-Côté, Dan Klein, and Ben Taskar. An end-to-end discriminative approach to machine translation. In *Proceedings of the International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*, 2006.

[99] Dekang Lin and Patrick Pantel. Dirt - discovery of inference rules from text. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 323–328, 2001.

[100] Thomas Lin, Mausam, and Oren Etzioni. Identifying functional relations in web text. In *Proceedings of the Conference on Empirial Methods in Natural Language Processing (EMNLP)*, pages 1266–1276, 2010.

[101] Greg Little and Robert C. Miller. Translating Keyword Commands into Executable Code. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 135–144, 2006.

[102] Jennifer Mankoff, Scott E. Hudson, and Gregory D. Abowd. Interaction techniques for ambiguity resolution in recognition-based interfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 11–20, 2000.

[103] Marie-Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2006.

[104] Andrew McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 359–367, 1998.

[105] Andrew McCallum, Karl Schultz, and Sameer Singh. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Proceedings of the Conference on Neural Information Processing Systems Conference (NIPS)*, 2009.

[106] Robert McCann, Warren Shen, and AnHai Doan. Matching schemas in online communities: A web 2.0 approach. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 110–119, 2008.

[107] Ryan T. McDonald, Fernando C. N. Pereira, Seth Kulick, R. Scott Winters, Yang Jin, and Peter S. White. Simple algorithms for complex relation extraction with applications to biomedical ie. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

[108] Luke McDowell and Michael J. Cafarella. Ontology-driven information extraction with ontosyphon. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 428–444, 2006.

[109] Daniel C. McFarlane. Comparison of four primary methods for coordinating the interruption of people in human-computer interaction. *Human-Computer Interaction*, 17(1):63–139, 2002.

[110] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[111] Scott Miller, Jethran Guinness, and Alex Zamanian. Name tagging with word clusters and discriminative training. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2004.

[112] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1003–1011, 2009.

[113] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM)*, pages 227–236, 2011.

[114] Joakim Nivre and Jens Nilsson. Memory-based dependency parsing. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 49–56, 2004.

[115] Tim Paek, Susan T. Dumais, and Ron Logan. WaveLens: A New View onto Internet Search Results. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, pages 727–734, 2004.

[116] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Libraries, 1998.

[117] Patrick Pantel and Marco Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the Annual Meetings of the Association for Computational Linguistics (ACL)*, 2006.

[118] Marius Pasca. Outclassing wikipedia in open-domain information extraction: Weakly-supervised acquisition of attributes over conceptual hierarchies. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 639–647, 2009.

[119] Hoifung Poon and Pedro Domingos. Joint inference in information extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 913–918, 2007.

[120] Hoifung Poon and Pedro Domingos. Unsupervised semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–10, 2009.

[121] Hoifung Poon and Pedro Domingos. Unsupervised ontology induction from text. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 296–305, 2010.

[122] Reid Priedhorsky, Jilin Chen, Shyong K. Lam, Katherine A. Panciera, Loren G. Terveen, and John Riedl. Creating, destroying, and restoring value in wikipedia. In *Proceedings of the ACM Conference on Supporting Group Work (GROUP)*, pages 259–268, 2007.

[123] Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher D. Manning. A multi-pass sieve for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 492–501, 2010.

[124] Cartic Ramakrishnan, Krys Kochut, and Amit P. Sheth. A framework for schema-driven relationship discovery from unstructured text. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 583–596, 2006.

[125] Deepak Ravichandran and Eduard H. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 41–47, 2002.

[126] Frederick Reiss, Sriram Raghavan, Rajasekar Krishnamurthy, Huaiyu Zhu, and Shivakumar Vaithyanathan. An algebraic approach to rule-based information extraction. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 933–942, 2008.

[127] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 148–163, 2010.

[128] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 148–163, 2010.

[129] Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 811–816, 1993.

[130] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *AAAI/IAAI, Vol. 2*, pages 1044–1049, 1996.

[131] Alan Ritter, Mausam, and Oren Etzioni. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 424–434, 2010.

[132] Dan Roth and Wen-Tau Yih. A Linear Programming Formulation for Global Inference in Natural Language Tasks. In *Proceedings of the 2004 Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8, 2004.

[133] Horacio Saggion, Adam Funk, Diana Maynard, and Kalina Bontcheva. Ontology-based information extraction for business intelligence. In *Proceedings of the International Semantic Web Conference (ISWC) and Asian Semantic Web Conference (ASWC)*, pages 843–856, 2007.

[134] Gerard Salton, James Allan, and Chris Buckley. Automatic structuring and retrieval of large text files. *Communications of the ACM*, 37(2):97–108, 1994.

[135] Gerard Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[136] Evan Sandhaus. *The New York Times Annotated Corpus*. Linguistic Data Consortium, 2008.

[137] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2004.

[138] Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[139] Warren Shen, AnHai Doan, Jeffrey F. Naughton, and Raghu Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *In Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 1033–1044, 2007.

[140] Michael Shilman, Desney S. Tan, and Patrice Simard. Cuetip: a mixed-initiative interface for correcting handwriting errors. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 323–332, 2006.

[141] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computation Linguistics (HLT-NAACL)*, 2006.

[142] Ben Shneiderman, Catherine Plaisant, Maxine Cohen, and Steven Jacobs. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, 2010.

[143] Parag Singla and Pedro Domingos. Entity Resolution with Markov Logic. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 572–582, 2006.

[144] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2004.

[145] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263, 2008.

[146] Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 129–136, 2011.

[147] Stephen Soderland, David Fisher, Jonathan Aseltine, and Wendy G. Lehnert. Crystal: Inducing a conceptual dictionary. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1314–1321, 1995.

[148] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A large ontology from wikipedia and wordnet. *Elsevier Journal of Web Semantics*, 6(3):203–217, 2008.

[149] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. Sofie: A self-organizing framework for information extraction. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2009.

[150] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2012.

[151] Partha Pratim Talukdar, Thorsten Brants, Mark Liberman, and Fernando Pereira. A context pattern induction method for named entity extraction. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, 2006.

[152] Partha Pratim Talukdar, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 582–590, 2008.

[153] Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 406–414, 1999.

[154] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press, 1988.

[155] Paul A. Viola and Mukund Narasimhan. Learning to extract information from semi-structured text using a discriminative context free grammar. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 330–337, 2005.

[156] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, pages 319–326, 2004.

[157] Luis von Ahn and Laura Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.

[158] Jakob Voss. Measuring wikipedia. In *Proceedings of the International Conference of the International Society for Scientometrics and Informetrics (ISSI)*, pages 221–231, 2005.

[159] Daisy Zhe Wang, Michael J. Franklin, Minos N. Garofalakis, Joseph M. Hellerstein, and Michael L. Wick. Hybrid in-database inference for declarative information extraction. In *SIGMOD Conference*, pages 517–528, 2011.

[160] Daisy Zhe Wang, Eirinaios Michelakis, Michael J. Franklin, Minos N. Garofalakis, and Joseph M. Hellerstein. Probabilistic declarative information extraction. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 173–176, 2010.

[161] Daisy Zhe Wang, Long Wei, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan. Selectivity estimation for extraction operators over text data. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 685–696, 2011.

[162] Gang Wang, Yong Yu, and Haiping Zhu. Pore: Positive-only relation extraction from wikipedia text. In *Proceedings of the International Semantic Web Conference (ISWC) and Asian Semantic Web Conference (ASWC)*, pages 580–594, 2007.

[163] Richard C. Wang and William W. Cohen. Language-independent set expansion of named entities using the web. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 342–350, 2007.

[164] Richard C. Wang and William W. Cohen. Iterative set expansion of named entities using the web. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2008.

[165] Ye-Yi Wang, Raphael Hoffmann, Xiao Li, and Alex Acero. Semi-supervised acquisition of semantic classes – from the web and for the web. In *International Conference on Information and Knowledge Management (CIKM)*, pages 37–46, 2009.

[166] Christopher A. Welty and J. William Murdock. Towards knowledge acquisition from information extraction. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 709–722, 2006.

[167] Fei Wu, Raphael Hoffmann, and Daniel S. Weld. Information extraction from wikipedia: moving down the long tail. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 731–739, 2008.

[168] Fei Wu and Daniel S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 41–50, 2007.

[169] Fei Wu and Daniel S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 635–644, 2008.

[170] Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *The Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 118–127, 2010.

[171] Limin Yao, Sebastian Riedel, and Andrew McCallum. Collective cross-document relation extraction without labelled data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1013–1023, 2010.

[172] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti A. Hearst. Faceted metadata for image search and browsing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, pages 401–408, 2003.

[173] Luke Zettlemoyer and Michael Collins. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, 2007.

[174] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2003.

[175] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 912–919, 2003.