Markov Logic for Machine Reading

Hoifung Poon

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

Doctor of Philosophy

University of Washington

2011

Program Authorized to Offer Degree: Computer Science and Engineering

University of Washington
Graduate School

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

Hoifung Poon

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

Chair of the Supervisory Committee:

Pedro Domingos

Reading Committee:

Pedro Domingos

Raymond J. Mooney

Luke Zettlemoyer

Date:

University of Washington

Abstract

Markov Logic for Machine Reading

Hoifung Poon

Chair of the Supervisory Committee:
Professor Pedro Domingos
Computer Science and Engineering

A long-standing goal of AI and natural language processing (NLP) is to harness human knowledge by automatically understanding text. Known as *machine reading*, it has become increasingly urgent with the rise of billions of web documents. However, progress in machine reading has been difficult, due to the combination of several key challenges: the complexity and uncertainty in representing and reasoning with knowledge, and the prohibitive cost in providing direct supervision (e.g., designing the meaning representation and labeling examples) for training a machine reading system.

In this dissertation, I propose a unifying approach for machine reading based on Markov logic. Markov logic defines a probabilistic model by weighted first-order logical formulas. It provides an ideal language for representing and reasoning with complex, probabilistic knowledge, and opens up new avenues for leveraging indirect supervision via *joint inference*, where the labels of some objects can be used to predict the labels of others.

I will demonstrate the promise of this approach by presenting a series of works that applied Markov logic to increasingly challenging problems in machine reading. First, I will describe a joint approach for citation information extraction that combines information among different citations and processing stages. Using Markov logic as a representation language and the generic learning and inference algorithms available for it, our solution largely reduced to writing appropriate logical formulas and was able to achieve state-of-the-art accuracy with substantially less engineering effort compared to previous approaches.

Next, I will describe an unsupervised coreference resolution system that builds on Markov logic to incorporate prior knowledge and conduct large-scale joint inference. This helps compensate for the lack of labeled examples, and our unsupervised system often ties or even outperforms previous state-of-the-art supervised systems.

Finally, I will describe the USP system, the first unsupervised approach for jointly inducing a meaning representation and extracting detailed meanings from text. To resolve linguistic variations for the same meaning, USP recursively clusters expressions that are composed with or by similar expressions. USP can also induce ontological relations by creating abstractions to assimilate commonalities among non-synonymous meaning clusters. This results in a state-of-the-art end-to-end machine reading system that can read text, extract knowledge and answer questions, all without any labeled examples. Markov logic provides an extremely compact representation of the USP model, and enables future work to "close the loop" by incorporating the extracted knowledge into the model to aid further extraction.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am fortunate to have Pedro as my advisor. Pedro showed me by his own example how to do great research - by committing to big goals while maintaining agility in seeking the path to get there. I am always inspired by his curiosity and energy, his erudition and insight, and his humor and optimism. I am also very lucky to have a stellar committee. Ray is both a great role model in research and an invigorating companion in travel. I hope that one day I could accomplish something that would make him exclaim: "it shatters my expectation!" Oren's sharp comments are always refreshing and his insistence on having real-world impact is illuminating. This dissertation was shaped in many ways by his pioneering work in open IE. Chris introduced me to the wonderland of NLP, through our first meeting back in 2006, through his excellent textbook and great research, and through his insightful remarks and constant encouragement. Luke's Best Paper in UAI-05 attracted me to the problem of semantic parsing, and I first learned about literature-based discovery from Meliha's dissertation. They are both warm and helpful whenever I am in need of guidance and advice.

Over the years, many people help shape my taste and direction in research. Andrew McCallum served in my generals exam committee and his insightful comments on joint inference motivated me to focus on unsupervised learning in subsequent works. The fantastic works by Aria Haghighi and Dan Klein made me see the potential in unsupervised learning and helped shape my way of approaching it. I am also much indebted to Dan Roth, Jude Shavlik, Dan Jurafsky, David McAllester, and many others for the inspirations I draw from their excellent research programs and for their generous advices. I had two memorable internships at Microsoft Research, which introduced me to an impressive array of excellent researchers, and led to very pleasant and productive collaborations with Colin Cherry, Kristina Toutanova, and Lucy Vanderwende.

UW-CSE offers a fantastic environment that balances work with fun. The department is uniquely sociable, the faculty are incredibly supportive and accessible, and Ed Lazowska's "spams" provides a noble excuse to procrastinate. Needless to say, friends make this world awesome. In particular, I want to thank Fei Wu, Yongchul Kwon, Parag Singla, Daniel Lowd, Jesse Davis, Vibhav Gogate, Scott Saponas, Jonathan Lester, Eytan Adar, Mike Cafarella, Noah Snavely, Chris Re, Krzysztof Gajos, Lin Liao, Colin Zheng, Xiao Ling, Peng Dai, Ning Chen, Xu Miao, Mausam, Mike Piatek, Suporn Pongnumkul, Nodira Khoussainova, Indriyati Atmosukarto.

My academic journey wouldn't be possible at all without the steadfast support of my family. My parents are always supportive and understanding. Rufang encouraged me to pursue my dream, shoulders an unfair share of family responsibilities, cheers me up in depressing times, and offers sobering critiques when I can't help gloating. Julia and Alice keep me sufficiently distracted to maintain sanity, and provide stellar examples of energy and optimism. This dissertation is dedicated to them.

DEDICATION

to my family

Chapter 1
INTRODUCTION

1.1 Machine Reading: The Promise and Challenges

A long-standing goal in AI and natural language processing (NLP) is to automate language understanding by extracting formal meaning from text [2].[1] Known as machine reading [25], it has become increasingly urgent with the advent of the web, which makes available billions of documents and virtually unlimited amount of knowledge [18]. Success in machine reading will not only help breach the knowledge acquisition bottleneck in AI, but also promise to revolutionize scientific and applied research, web search, and other fields.

For example, the PubMed online repository contains over 18 million abstracts on biomedical research, with more than two thousand new abstracts added each day. Such explosive growth of information makes it imperative to automate the scientific discovery processes [26]. By automatically extracting and synthesizing knowledge from research literature, machine reading can play a pivotal role in this direction.

Despite the promise, progress in machine reading has been difficult. In particular, three major challenges stand out:

- **Complexity:** Knowledge is highly heterogeneous, ranging from facts on individual objects (e.g., Socrates) and general types (e.g., man), to generic inference rules (e.g., syllogism) and ontological relations (e.g., ISA and ISPART). In addition, language processing is very complex and requires many challenging and interdependent subtasks such as morphological analysis, part-of-speech tagging, syntactic chunking and parsing, coreference resolution, semantic parsing, etc.

- **Uncertainty:** Natural languages are highly ambiguous, and a lot of knowledge is

[1]By "formal" we mean that a canonical meaning representation is in use, which facilitates querying and reasoning, and can potentially be actionable.

about soft correlation rather than hard constraints. Modeling of the world is always incomplete and therefore predictions are inherently uncertain.

- **Long tail of variations:** Natural languages contain myriad variations in expressing the same meaning. For example, "Microsoft buys Farecast" can be stated alternatively as "Farecast was bought by the Redmond software giant", "Microsoft's acquisition of Farecast", etc. Such variations pose serious challenges to machine reading systems, which can be visualized in Figure 1.1. The horizontal axis represents potential knowledge extractions, whereas the vertical axis represents how confident we are whether an extraction is correct or not. For well-known facts like Edison invented the practical light bulb, many independent occurrences exist, and it is relatively easy to extract them with high confidence. However, language variations mean that there is also an extreme long tail of extractions with few occurrences, which make it very difficult to assess their correctness.

Early approaches to machine reading were manual, but the sheer amount of coding and knowledge engineering needed makes them very costly and limits them to well-circumscribed domains. More recently, machine learning approaches dominate. However, existing learning approaches are inadequate in addressing all three challenges. Statistical learning methods typically circumvent the complexity in knowledge representation by focusing on subtasks that can be reduced to classification, where objects are modeled by i.i.d. feature vectors. The complexity in language processing is tackled by adopting a pipeline architecture, where the output of each stage is the input of the next, and there is no feedback from later stages to earlier ones. Although this makes the systems comparatively easy to assemble, and helps to contain computational complexity, it comes at a high price: errors accumulate as information progresses through the pipeline, and an error once made cannot be corrected. Moreover, the prevailing learning paradigm is supervised, which requires labeled data in the form of example input-output pairs. While raw text is available in virtually unlimited amount, labeled resources are scarce and the cost to build them is very high.[2] For large-

[2]For example, the development of the widely used and invaluable English Penn treebank for syntactic parsing [71] spanned a decade and yet the coverage is still quite limited.

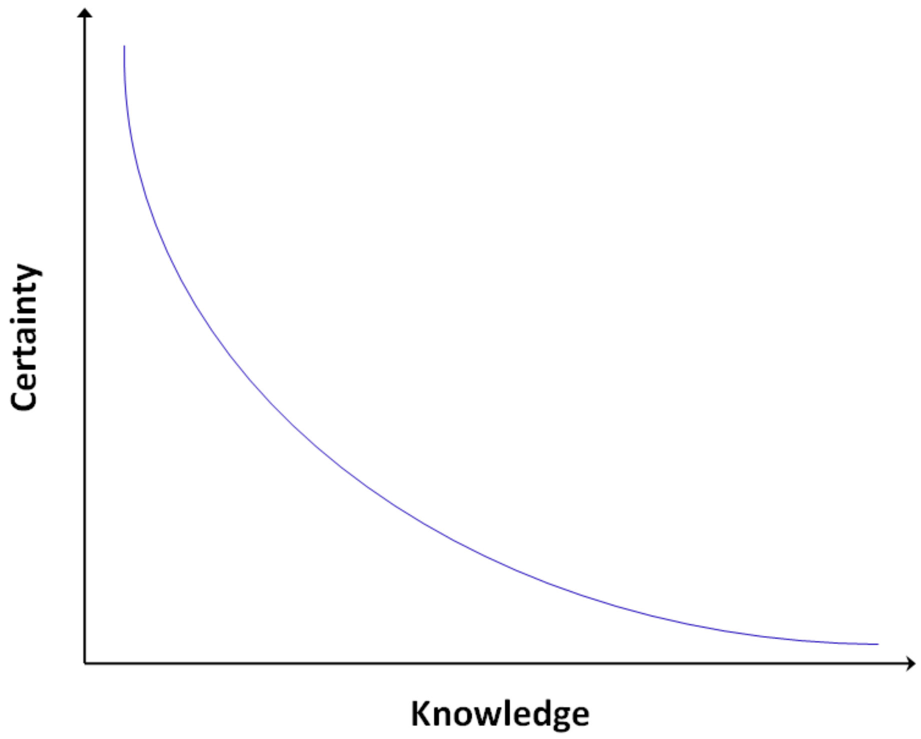

Figure 1.1: The long tail of variations: most potential knowledge extractions have few occurrences, which makes it very hard to determine whether they are correct or not.

scale, open-domain machine reading, the cost to acquire sufficient labeled examples for handling the long tail of variations is simply prohibitive.

1.2 Related Work

Inspired by the web and potential applications, interest in machine reading is rising quickly, as evidenced by the recent symposiums, workshops, and DARPA programs.[3] Existing work tends to focus on various subtasks of machine reading. For example, a lot of progress has been made in automating the components in the language processing pipeline, such as part-of-speech tagging [108], syntactic parsing [85], information extraction [73], coreference resolution [79], etc. The Never-Ending-Language-Learning (NELL) system starts from an

[3]E.g., AAAI-07 Machine Reading Symposium, NAACL-10 Learning by Reading Workshop, DARPA Machine Reading Program (2009-2014).

ontology with a few example instances for each concept, and learns to extract more instances by reading new text [12]. There is much work in textual entailment, which aims to determine if a proposition can be inferred from a small paragraph of text [20].

On the other hand, end-to-end solutions for machine reading are still rare, and existing systems typically require substantial amount of human effort in manual engineering and/or labeling examples. As a result, they often focus on restricted domains and limited types of extractions (e.g., a prespecified relation). A notable exception is the TextRunner system [8], which conducts open-domain machine reading by extracting relational tuples from raw text. Extraction is not limited to predefined relations and does not require labeled examples.

The TextRunner system and the "open information extraction" direction it pioneers signify a shift from the traditional paradigm that relies on direct supervision. For web-scale, open-domain machine reading, the cost to garner sufficient direct supervision is prohibitive. Consequently, there has been increasing interest in harnessing various sources of indirect supervision. For example, TextRunner first employs self-supervision to label noisy examples using a small set of hand-picked generic relational patterns, and then trains an extractor using standard supervised-learning methods [8, 7]. NELL uses an approach similar to bootstrapping [1]: it initializes the extractor with seed examples and then repeatedly labels new examples and retrains the extractor, with a novel addition that enforces the mutual-exclusive constraint among non-overlapping concepts and substantially improves extraction accuracy. To resolve language variations, a well-known idea is to leverage *distributional similarity* [40]: if two expressions often occur in similar contexts, they probably mean the same. For example, the typical approach in lexical semantics resolves synonymous words by clustering their contextual profiles (e.g., dependency neighbors [65]). Similarly, the DIRT system [67] resolves synonymous binary relations (as represented by dependency paths) using distributional information of the argument words.

The combined challenges of complexity and uncertainty are not only present in language understanding but also widespread in other areas of AI. In response, an emerging direction in machine learning aims to combat head-on with both challenges [34, 6]. Known as *statistical relational learning* and many other names (e.g., structured prediction, collective classification, multi-relational data mining, etc.), this movement has several salient agenda:

- **Joint inference:** The complexity challenge stems from the fact that real-world objects often have elaborate structures (e.g., the parse tree of a sentence, the part-decomposition of human body) and the objects and processing stages are interdependent. Joint inference aims to model such structures and interdependencies and incorporate them in learning and prediction.

- **Combine logic and probability:** First-order logic can succinctly represent complex dependencies, probability can handle uncertainty. Combining the two thus offers a powerful language to deal with the combination of complexity and uncertainty.

- **Automate the design of features and representations:** Much of the current success of machine learning relies on feature engineering, which requires domain expertise and substantial manual effort. As we enter the "big data" era, the ever rising scale and heterogeneity of the data make progress in this paradigm increasingly difficult. As a result, there has been increasing interest in automating the process of designing representations and features for modeling and learning. Examples include predicate invention [77, 53], structure learning [52], deep learning [62], etc.

- **Develop efficient algorithms for joint inference and learning:** The power in combining logic and probability comes at a price: inference becomes intractable, and learning becomes much harder since it requires inference as a subroutine. For statistical relational learning to materialize its promise, we need to develop a generic framework that makes it as easy to use and as efficient as traditional statistical learning.

1.3 A Unifying Approach to Machine Reading

Ideally, a machine reading system should satisfy the following desiderata [91]:

- **End-to-end:** the system should input raw text, extract knowledge, and be able to answer questions and support other end tasks;

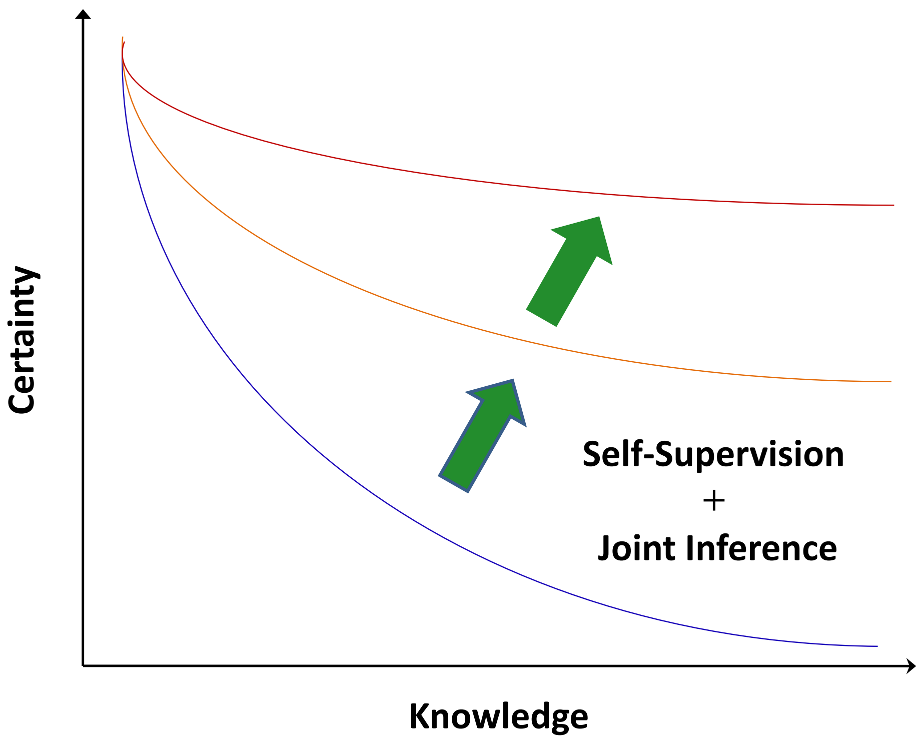

Figure 1.2: A vision for machine reading: bootstrap from the easily extractable knowledge, and conquer the long tail using a self-supervised learning process that propagates information from more certain extractions to less certain ones via joint inference.

- **High quality:** the system should extract knowledge with high accuracy;
- **Large-scale:** the system should require knowledge at web-scale and be open to arbitrary domains, genres, and languages;
- **Maximally autonomous:** the system should incur minimal human effort;
- **Continuous learning from experience:** the system should constantly integrate new information sources (e.g., new text documents) and learn from user questions and feedback (e.g., via performing end tasks) to keep improving its performance.

These desiderata raise many intriguing research questions. For example, how can a machine reading system overcome the limitation of pipeline architectures and enable information to propagate across processing stages? How can it seamlessly integrate information from heterogeneous sources? How can it incorporate extracted knowledge back to the model, thus closing the loop and continuously improving reading accuracy? How can it breach the bottleneck of labeled examples? What indirect supervision can be leveraged and how? How can the reading system adapt to different domains with minimal domain-specific engineering? How can all these be done while inference and learning remain efficient? How can such a system scale to the web?

In this dissertation, we propose a unifying approach to machine reading based on statistical relation learning. A distinctive focus of our approach is using joint inference to compensate for the lack of labeled examples. The key idea is that, by modeling interdependencies among objects, the labels of some objects can be used to predict the labels of others. Figure 1.2 shows the vision of our approach. By leveraging interdependencies among extractions, the system can bootstrap from easily extractable knowledge, and conquer the long tail by propagating information from more certain extractions to less certain ones. This process is governed by self-supervised learning which determines, among other things, what direct and indirect supervision to use and how.

To materialize this vision, there are several challenges to overcome. First, we need to develop a representation language. Clearly, such a language must be capable of handling uncertainty and must be expressive enough to accommodate a wide variety of direct and indirect supervision. In particular, a key source of supervision can derive from world knowledge, which is crucial for resolving ambiguities in language understanding [2]. Therefore, the ideal representation language should make it easy to incorporate arbitrary knowledge into the model, thus closing the loop between the extraction process and the knowledge output.

We propose to use Markov logic [23] as our representation language for machine reading. Markov logic defines a probability distribution using weighted first-order logical formulas as feature templates. First-order logic can compactly encode very complex dependencies, and has been extensively studied for knowledge representation. The main drawback is that logic is deterministic and can not handle noise and uncertainty. This limitation is overcome in Markov logic by softening the formulas with weights to account for uncertainty. Extracted knowledge can be naturally represented as weighted formulas, which can be added back to the model to guide future extractions.

Second, we need to bridge the schism between knowledge representation (KR) and NLP. The two fields used to recognize each other as complementary for the common goal of machine reading: KR is responsible for designing the formal system to represent and reason with knowledge, whereas NLP is responsible for converting text into the given canonical meaning representation after resolving variations and ambiguities. Unfortunately, the marriage was short-lived and this vision went into oblivion [14]. The KR community become preoccupied with devising and revising numerous logical formalisms for knowledge and belief, without worrying much about NLP and how the knowledge comes by. As a result, the formalisms so designed are often not very useful in NLP. In particular, the logical forms are typically disconnected from text and there requires substantial effort to map between the two [110]. Meanwhile, the NLP community, disillusioned by the lack of progress in this direction, largely abandon the vision and sidestep the KR problem by focusing on subtasks that do not require sophisticated knowledge structures. There remain attempts to extract deep meanings from text (e.g., [76, 119]), but they presuppose that the meaning representation has been manually designated and example text-meaning pairs are available for learning the mapping. As mentioned earlier, the long tail of variations render both assumptions questionable in open-domain machine reading.

We propose to reunite KR and NLP by automatically inducing knowledge representation from text. This would benefit from joint inference among the two endeavors, and solve the core problem in the original divide-and-conquer scheme: the disconnect between text and representation. By automating the KR process, this approach helps reduce manual effort and overcome the lack of direct supervision. For statistical relational learning, the unification of KR and NLP raises intriguing challenges and offers an ideal "killer app".

Third, for web-scale machine reading, it is imperative to develop efficient algorithms for joint inference and learning that can scale up to the whole web. *To address this challenge, we propose to adopt a new paradigm of probabilistic modeling by incorporating computation into the model and automating the trade-off between modeling fidelity and inference efficiency.* Dynamic programming is a particular effective paradigm for efficient inference and is widely used in NLP, vision, and other fields (e.g., [43, 28]). However, existing approaches typically adopt a specific trade-off between efficiency and accuracy by prespecifying the structure for dynamic programming (e.g., linear chains in HMMs, charts in PCFG parsing). By automatically inducing the dynamic programming structure, we can let data to decide where it is necessary to model the dependencies and where independence assumptions can be made without compromising the accuracy by much, while ensuring efficient inference at performance time.

To scale up to the web, we propose to adopt coarse-to-fine inference and learning [29, 84, 47]. Essentially, coarse-to-fine inference leverages the sparsity imposed by hierarchical structures that are ubiquitous in human knowledge (e.g., taxonomies/ontologies). At coarse levels, ambiguities are rare (there are few objects and relations), and inference can be conducted efficiently. The results are then used to prune unpromising refinements at the next level. This process continues down the hierarchy until decision can be made. In this way, inference can potentially be sped up exponentially, analogous to binary search tree.

The success of coarse-to-fine inference hinges on the availability of appropriate hierarchical structures. An ontology specifies entities and their relations in a problem domain, among which are the ISA and ISPART hierarchies that can be used to support coarse-to-fine inference. In general, constructing the ontology (ontology induction) and mapping textual expressions to ontological nodes (ontology population) remain difficult open problems [103]. *We propose to automatically induce ontology from text, with supporting efficient coarse-to-fine inference being a core component of the learning objective.* In general, we hypothesize that computational constraints offer a potent bias for learning knowledge representation (also see our second proposed agenda above).[4]

Finally, we observe that there are inherent problems in existing evaluation methodologies. In the standard paradigm, the system output is compared with gold annotation by an intrinsic measure. There are two problems with this approach. First, it requires an annotated corpus which is expensive and time-consuming to construct. Second, the evaluation is

[4]We may even ask a broader question: why do humans invent the symbolic representations such as concepts, relations, taxonomies? It is appealing to regard them as mental tools for tackling the complexity of the real world by facilitating efficient modeling, communication, and decision making. A thorough exploration in this question is fascinating but is beyond the scope of this dissertation.

biased to the specific annotation formalism in use and may bear little indication to the real contribution for an end-to-end task. Examples include word sense disambiguation, semantic role labeling, and word alignment. Intuitively, progress in these subtasks will clearly help end applications. In practice, it is non-trivial to materialize the gains [115], and sometimes the payoff is quite small [38].

We propose to develop a new evaluation methodology for machine reading that includes end users and applications in the loop. The knowledge output from machine reading can be used in end services such as question answering and decision making. By designing an appropriate mechanism, we can motivate end users to use the services while deriving explicit or implicit feedback that can be used to evaluate the machine reading system as well as provide learning signals for continuously improving reading accuracy.

In sum, we propose a unifying approach for machine reading as follows:

- Adopt Markov logic as the unifying framework for knowledge representation and joint inference.

- Bootstrap from easily extractable knowledge and conquer the long tail of language variations via a self-supervised learning process that incorporates knowledge with large-scale joint inference and closes the loop between the extraction process and knowledge output.

- Scale up joint inference with coarse-to-fine inference.

- Incorporate computation into probabilistic modeling for learning the knowledge representation. In particular, automatically induce probabilistic ontologies from text for better generalization and efficient coarse-to-fine inference.

- Accomplish continuous learning by combining bootstrapping and crowdsourced content creation to synergistically improve the reading system from user interaction.

1.4 Overview of this Dissertation

In this dissertation, we take some initial steps following this approach and show that it is quite promising.

In Chapter 2, we review the theory and algorithms for Markov logic, which form the basis for the remaining chapters.

In Chapter 3, we apply Markov logic to learn a joint model for information extraction in the citation domain. We show that Markov logic makes it easier to perform joint inference: it can compactly specify a very large and complex probabilistic model, and attain state-of-the-art accuracy using the generic learning and inference algorithms for it.

In Chapter 4, we develop an unsupervised weight learning algorithm for Markov logic and apply it to coreference resolution. We show that Markov logic can have even greater impact in the more challenging setting where labeled examples are not available: using joint inference and a small amount of prior knowledge, our unsupervised system often ties or even outperforms previous supervised systems.

In Chapter 5, we take an initial step toward automating the development of knowledge representation: we propose the first unsupervised approach for semantic parsing and develop the USP system, which automatically induces a canonical meaning representation based on Markov logic by recursively clustering language variations for the same meaning. Additionally, we propose a novel evaluation method that applies semantic parsing to the end task of reading text and answering questions. We show that by jointly identifying meaning units and clustering variations, USP substantially improves the accuracy and recall compared to state-of-the-art systems such as TextRunner.

In Chapter 6, we extend USP to also induce ontological relations among the meaning clusters. We show that this can drastically improve the recall in question answering without affecting the accuracy, and that learning can be made possible by assimilating commonalities such as similar argument types among meaning clusters.

We conclude in Chapter 7 with a discussion on the limitations of this work and exciting future directions.

Many of the datasets and code in this dissertation are available online at `alchemy.cs.washington.edu`. In particular, for information extraction and unsupervised coreference resolution, we used standard datasets and extensions to the Alchemy open-source software. The code and dataset for unsupervised semantic parsing are available at `alchemy.cs.washington.edu/usp`; see Chapter 5 for details.

Chapter 2
MARKOV LOGIC

2.1 Introduction

Markov logic is a probabilistic extension of finite first-order logic [94, 23].[1] Logic can compactly specify elaborate relations among many objects. However, it is deterministic and therefore brittle in the presence of noise and uncertainty. The key idea of Markov logic is to overcome this brittleness by softening the formulas with weights, thereby defining a probability distribution as a log-linear model.

Markov logic provides as an interface layer and enables a healthy separation between foundational work and applications [23]. Advances in inference and learning algorithms can immediately benefit numerous applications; application developers can concentrate on application-specific modeling without worrying much about algorithms and implementation. Such an interface layer enables faster overall progress, better reuse of algorithmic ideas and implementations, and substantially reduced engineering effort in applications.

Among the numerous approaches proposed for statistical relational learning [34], Markov logic stands out as a leading unifying framework. Despite its simplicity, Markov logic is very general, subsuming both first-order logic and probabilistic graphical models. In comparison, other approaches typically use a restricted subset of first-order logic (e.g., database query [107], Horn theory [113, 80, 46]), which limits the types of dependencies and uncertainty that can be expressed. This also means that other approaches can be straightforwardly converted into Markov logic, which makes it easy to compare approaches, share lessons and advances, and assemble existing modules into a larger system.

Compared to other approaches, Markov logic is arguably the most developed. There are algorithms available for major inference and learning tasks, ranging from MAP and marginal inference, to learning weights, formulas, and even predicates.

[1] Extensions to infinite domains exist, such as [100].

Open-source software implementations are available for existing algorithms of Markov logic, with Alchemy [57] being the most complete to this date. There is a growing community of Markov logic[2] and many successful applications.

2.2 Background

First-order logic provides a precise language for compactly specifying a complex domain with many objects [33]. A *term* is a variable or constant representing an object in the domain, such as `Anna` and `x`. An *atomic formula* or *atom* is a predicate symbol applied to a tuple of terms that describes relations, such as $\texttt{Friends}(\texttt{x},\texttt{y})$, $\texttt{Smoke}(\texttt{Anna})$. Formulas are recursively constructed from atomic formulas using logical connectives $(\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow)$ and quantifiers $(\forall, \exists)$. Below are two example formulas.

$$\forall \texttt{x} \;\; \texttt{Smoke}(\texttt{x}) \Rightarrow \texttt{Cancer}(\texttt{x})$$

$$\forall \texttt{x} \;\; \texttt{Friend}(\texttt{x},\texttt{y}) \Rightarrow (\texttt{Smoke}(\texttt{x}) \Leftrightarrow \texttt{Smoke}(\texttt{y}))$$

The first formula says that if `x` smokes, then `x` has cancer; the second formula says that if $\texttt{x},\texttt{y}$ are friends, then if `x` smokes, `y` smokes, and vice versa. A *first-order knowledge base (KB)* is a set of first-order logical formulas. A *ground term* is a term containing no variables. A *ground atom* or *ground predicate* is an atom all of whose arguments are ground terms. A *ground formula* is a formula that contains only ground atoms.

To handle uncertainty, probability is the method of choice. A popular formalism for probabilistic modeling is graphical model, which can compactly specify probability distributions by exploiting conditional independence [83, 58]. A *Markov network* (a.k.a. Markov random field) is an undirected graphical model that defines a joint distribution over a set of variables $X = (X_1, \cdots, X_n)$ [83] as follows:

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}) \tag{2.1}$$

where ϕ_k is a *potential function* defined on a subset of variables, and $x_{\{k\}}$ is the state of the variables in that subset. $Z = \sum_{x \in \mathcal{X}} \prod_k \phi_k(x_{\{k\}})$ is the normalization constant, also known

[2]By September 2009, Alchemy had been downloaded for 5000 times. As of June 2011, the number has reached 9000.

as the *partition function.* The compactness of this model derives from the factorization into potential functions, each of which ideally involves only a small number of variables. Defining a potential function requires as many entries as the total number of states for the corresponding subset of variables. But this can be drastically simplified by leveraging redundancy in the entries and adopting a feature representation. Consequently, Markov networks are often represented as *log-linear models*, with each potential replaced by an exponentiated weighted sum of features of the state as follows:

$$P(X=x) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(x)\right) \tag{2.2}$$

2.3 Representation

Definition 1. *A* Markov logic network (MLN) *is a set of weighted first-order formulas. Together with a set of constants, it defines a Markov network with one variable per ground atom and one feature per ground formula. The weight of a feature is the weight of the first-order formula that originated it. The probability of a state x in such a network is given by*

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right),$$

where Z is the normalization constant as in Markov networks, w_i is the weight of the ith formula and $n_i(x)$ is the number of its true groundings given x.

Intuitively, the higher the weight, the stronger the regularity represented by the formula. With infinite weight, the formula becomes a logical constraint.[3] First-order logic is a special case with all weights being infinite. For Markov networks, Markov logic formulas define the feature templates and offer an elegant way to tie parameters.

As the number of objects increases, the size of the ground Markov network grows exponentially in the number of variables in a formula, whereas Markov logic maintains its

[3]In practice, it suffices to use a large number. For example, a weight of 20 makes the formula about 400 million times more likely to be true than false, everything else being equal.

compactness. Moreover, the first-order structure can be leveraged to improve efficiency in inference and learning (e.g., see lazy inference in the next section).

2.4 Inference

Given a probabilistic model, the inference problem is to compute the answer to a query. Specifically, maximum a posterior (MAP) inference finds the most probable state given the evidence (i.e., a partial truth assignment of the variables), whereas marginal inference computes the conditional probability for a formula to be true.

Like graphical models, inference in Markov logic is intractable [95]. Thus an important research agenda is to develop efficient methods for approximate inference. To do this, we can build on existing logical and statistical inference methods and piggyback on decades of prior research. For example, the MAP inference problem in Markov logic can be reduced to a weighted SAT problem, and can be solved using a weighted SAT solver such as MaxWalkSAT [44].

However, existing methods are often inadequate by themselves. For example, the standard approach for marginal inference is Markov chain Monte Carlo (MCMC), which samples from the states by constructing a Markov chain and computes the probabilities using the samples [35]. A popular MCMC algorithm is Gibbs sampling, which iteratively samples the next state of a variable while fixing the current value of other variables. When there are logical constraints, the state space is broken into disjoint regions and it is very hard to jump between regions using such a local update scheme. As a result, Gibbs sampling tends to get trapped in the initial region and return incorrect probabilities. This is the kind of challenges we face when we try to unify logic and probability.

Fortunately, we can turn the challenge into an opportunity by combining logical and statistical methods. For example, we developed the MC-SAT algorithm [87], which uses a SAT solver to propose a sample for MCMC. MC-SAT is a “slice sampling” MCMC algorithm. Slice sampling introduces auxiliary variables u that decouple the original ones x, and alternately samples u conditioned on x and vice-versa. To sample from the slice (the set of states x consistent with the current u), MC-SAT calls SampleSAT [112], which uses a combination of satisfiability testing and simulated annealing. The advantage of using a sat-

Algorithm 1 MC-SAT(*formulas, weights, num_samples*)

```
$x^{(0)} \leftarrow$ Satisfy(hard *formulas*)
for $i \leftarrow 1$ to *num_samples* do
    $M \leftarrow \emptyset$
    for all $c_k \in$ *formulas* satisfied by $x^{(i-1)}$ do
        With probability $1 - e^{-w_k}$ add $c_k$ to $M$
    end for
    Sample $x^{(i)} \sim \mathcal{U}_{SAT(M)}$
end for
```

isfiability solver (WalkSAT [97]) is that it efficiently finds isolated modes in the distribution, and as a result the Markov chain mixes very rapidly. The slice sampling scheme ensures that detailed balance is (approximately) preserved. MC-SAT is orders of magnitude faster than previous MCMC algorithms like Gibbs sampling, making efficient sampling possible on a scale that was previously out of reach.

Algorithm 1 gives pseudo-code for MC-SAT. At iteration $i-1$, the factor ϕ_k for clause c_k is either e^{w_k} if c_k is satisfied in $x^{(i-1)}$, or 1 otherwise. MC-SAT first samples the auxiliary variable u_k uniformly from $(0, \phi_k)$, then samples a new state uniformly from the set of states that satisfy $\phi'_k \geq u_k$ for all k (the slice). Equivalently, for each k, with probability $1 - e^{-w_k}$ the next state must satisfy c_k. In general, we can factorize the probability distribution in any way that facilitates inference, sample the u_k's, and make sure that the next state is drawn uniformly from solutions that satisfy $\phi'_k \geq u_k$ for all factors.

MC-SAT, like most existing relational inference algorithms, grounds all predicates and formulas, thus requiring memory and time exponential in the predicate and formula arities (i.e., the number of variables in the predicate or formula). We developed a general method for producing a "lazy" version of a wide variety of relational inference algorithms [89], which carries exactly the same inference steps as the original algorithm, but only maintains a small subset of "active" predicates/formulas and grounds more as needed. We show that lazy-MC-SAT, the lazy version of MC-SAT, reduced memory and time by orders of magnitude in several domains [89].

2.5 Learning

There are three major learning tasks for Markov logic.

- **Weight learning:** Given the formulas, learn their weights. This is the most common learning scenarios in existing applications. The formulas define the features, which are devised by domain experts and application developers. Learning amounts to finding optimal parameters for some learning objective such as maximizing the likelihood.

- **Structure learning:** Learn both the formulas and weights. The Markov logic network is initialized by an initial KB (which may be empty). The goal is to construct new formulas and/or refine old ones by learning from the data. This is a much more ambitious task that amounts to feature induction or feature learning.

- **Predicate invention:** In addition to learning formulas and weights, also learn to create new predicates by composing base predicates. This would make it possible to create formulas using the new predicates to compactly capture much more general regularities.

The learning scenario can also be classified into *supervised* and *unsupervised*, by whether the system makes use of labeled examples (i.e., example input-output pairs). If the system uses labeled examples as well as many unlabeled examples (i.e., only the input is available), the learning problem is *semi-supervised*.

Exist works in statistical relational learning and Markov logic are mostly supervised. They also tend to focus on weight learning, although progress has been made in structure learning as well (e.g., [52, 74, 55, 56]). Unsupervised works exist mainly in the form of *relational clustering*, which clusters relations and objects simultaneously (e.g., [45, 53, 54]). Relational clustering can also be viewed as a form of predicate invention, with the clusters forming new predicates.

For supervised weight learning, a standard learning objective is to maximize the conditional log-likelihood $L(x, y) = \log P(Y = y|X = x)$, where Y represents the non-evidence predicates, X the evidence predicates, and x, y their values in the training data. For simplicity, from now on we omit X, whose values are fixed and always conditioned on. The optimization problem is convex and a global optimum can be found using gradient descent, with the gradient being

$$\begin{aligned} \frac{\partial}{\partial w_i} L(y) &= n_i(y) - \sum_{y'} P(Y = y') n_i(y') \\ &= n_i(y) - E_Y[n_i]. \end{aligned}$$

where n_i is the number of true groundings of formula i in the training data and $E_Y[n_i]$ is the expected count given the current parameterization. To combat overfitting, a Gaussian prior is imposed on all weights. Here, the core challenge is computing the expected counts, which requires inference that is generally intractable. One solution is to use existing MCMC algorithms such as Gibbs sampling [94]. Namely, the expected count can be approximated as

$$E_Y[n_i] \approx \frac{1}{N} \sum_{k=1}^{N} n_i(y_k)$$

where y_k are the samples generated by Gibbs sampling. However, Gibbs sampling is often quite slow and may completely break down when hard constraints are present. An alternative approach, as exemplified by the voted percetron algorithm [17], is to approximate the expected counts using the MAP state. The original voted percetron algorithm was developed for hidden Markov models, where computing the MAP state can be done exactly using the Viterbi algorithm. It has since been generalized to Markov logic by replacing the Viterbi algorithm with MaxWalkSAT [98].

Another major challenge is the problem of *ill-conditioning*. In gradient descent, the ideal learning rate for each parameter can vary to a great deal. The ratio between the highest and lowest rates is called the *condition number*.[4] An optimization problem with condition number in the hundreds is considered to be quite challenging. In Markov logic, the ideal rate for the weight of a formula correlates with the number of groundings for the formula, which means that the condition number can easily rise to millions or even billions.

[4]More precisely, the condition number is the ratio between the largest and smallest absolute eigenvalues of the Hessian.

In this dissertation, we build on previous works and make new contributions to learning in Markov logic.

In Chapter 3, we describe a supervised weight learning algorithm that uses MC-SAT to estimate the gradient and uses a heuristic adjustment to combat the ill-conditioning problem.

In Chapter 4, we describe a general unsupervised weight learning algorithm also based on MC-SAT. It can also be used in semi-supervised learning if labeled examples are available.

In Chapter 5, we describe an algorithm for *recursive relational clustering*, a novel form of relational clustering that jointly segments the input into meaning units and clusters the units.

In Chapter 6, we extend the algorithm in Chapter 5 to form an ISA hierarchy.

Chapter 3
JOINT INFERENCE IN INFORMATION EXTRACTION

3.1 Introduction

Much existing work in machine reading focuses on *information extraction*, which aims to extract database records from text or semi-structured sources according to a prespecified schema. Traditional approaches to information extraction typically adopt a pipeline architecture: the output of each stage is the input of the next, and there is no feedback from later stages to earlier ones. Two key stages in the pipeline are *segmentation*, which locates candidate fields, and *entity resolution*, which identifies duplicate records. Typically, each candidate record or field is segmented separately, and the output of this process is then passed to the entity resolution stage. To illustrate the shortcomings of this approach, consider the problem of extracting database records from the following two citations in CiteSeer [61]:

Minton, S(1993 b). Integrating heuristics for constraint satisfaction problems: A case study. In: Proceedings AAAI.

S. Minton Integrating heuristics for constraint satisfaction problems: A case study. In AAAI Proceedings, 1993.

In the first citation, author and title are clearly separated by a date and period, and extracting them is fairly straightforward. In the second one, there is no clear author-title boundary, and correctly pinpointing it seems very difficult. Large quantities of labeled training data and an extensive lexicon could help, but they are expensive to obtain, and even then are far from a guarantee of success. However, if we notice that the two citations are coreferent and the title of the first one begins with the substring "Integrating heuristics for," we can hypothesize that the title of the second one also begins with this substring, allowing us to correctly segment it.

Ideally, we would like to perform *joint inference* for all relevant tasks simultaneously. However, setting up a joint inference model is usually very complex, and the computational cost of running it can be prohibitive. Further, joint inference can sometimes hurt accuracy, by increasing the number of paths by which errors can propagate. As a result, fully joint approaches are still rare, and even partly-joint ones require much engineering.

In this chapter we propose a joint approach to information extraction based on Markov logic, where segmentation of all records and entity resolution are performed together in a single integrated inference process. While a number of previous authors have taken steps in this direction (e.g., [82, 114]), to our knowledge this is the first fully joint approach. In experiments on the CiteSeer and Cora citation matching datasets, joint inference improved accuracy, and our approach outperformed previous ones. Further, by using Markov logic and the generic algorithms for it, our solution consisted mainly of writing the appropriate logical formulas, and required much less engineering than previous ones.

3.2 Related Work

A number of previous authors have taken steps toward joint inference in information extraction, but to our knowledge no fully joint approach has been proposed. For example, [10] applied joint segmentation to protein name extraction, but did not do entity resolution. In citation matching, Pasula et al. [82] developed a "collective" model, but performed segmentation in a pre-processing stage, allowing boundaries to occur only at punctuation marks. Wellner et al. [114] extended the pipeline model by passing uncertainty from the segmentation phase to the entity resolution phase (as opposed to just passing the "best guess"), and by including a one-time step from resolution to segmentation, but did not "close the loop" by repeatedly propagating information in both directions. (Due to this, they explicitly refrained from calling their model "joint.") Both of these models required considerable engineering. Pasula *et al.* combined several models with separately learned parameters, a number of hard-wired components, and data from a variety of sources (including a database of names from the 2000 US Census, manually-segmented citations, and a large AI BibTex bibliography). Wellner *et al.* assembled a number of different learning and inference algorithms and performed extensive feature engineering, including computing a variety of string edit distances, TF-IDF measures, global features of the citations, and combinations thereof. As we will see, our approach is considerably simpler, and outperforms both of these.

3.3 An MLN for Joint Citation Matching

Citation matching is the problem of extracting bibliographic records from citation lists in technical papers, and merging records that represent the same publication. It has been a major focus of information extraction research (e.g., [82], [114]). We use citation matching as a testbed for joint inference with Markov logic. In particular, we focus on extracting titles, authors and venues from citation strings. In this section we present an MLN for this task.[1] In the next section, we learn weights and perform inference with it. While the MLN is specific to citation matching, we believe that the key ideas in it are also applicable to other information extraction tasks (e.g., extraction from Web pages, or from free text).

The main evidence predicate in the MLN is $\mathtt{Token}(\mathtt{t}, \mathtt{i}, \mathtt{c})$, which is true iff token $\mathtt{t}$ appears in the $\mathtt{i}$th position of the $\mathtt{c}$th citation. A token can be a word, date, number, etc. Punctuation marks are not treated as separate tokens; rather, the predicate $\mathtt{HasPunc}(\mathtt{c}, \mathtt{i})$ is true iff a punctuation mark appears immediately after the $\mathtt{i}$th position in the $\mathtt{c}$th citation. The query predicates are $\mathtt{InField}(\mathtt{i}, \mathtt{f}, \mathtt{c})$ and $\mathtt{SameCitation}(\mathtt{c}, \mathtt{c}')$. $\mathtt{InField}(\mathtt{i}, \mathtt{f}, \mathtt{c})$ is true iff the $\mathtt{i}$th position of the $\mathtt{c}$th citation is part of field $\mathtt{f}$, where $\mathtt{f} \in \{\mathtt{Title}, \mathtt{Author}, \mathtt{Venue}\}$, and inferring it performs segmentation. $\mathtt{SameCitation}(\mathtt{c}, \mathtt{c}')$ is true iff citations $\mathtt{c}$ and $\mathtt{c}'$ represent the same publication, and inferring it performs entity resolution.

3.3.1 Isolated Segmentation

We begin by describing our standalone segmentation model. This will form part of the overall MLN for joint inference, and also serve as a baseline for comparison. Our segmentation model is essentially a hidden Markov model (HMM) with enhanced ability to detect field boundaries. This combines two key elements of the state of the art. The observation matrix of the HMM correlates tokens with fields, and is represented by the simple rule

[1]Available at http://alchemy.cs.washington.edu/papers/poon07.

$$\texttt{Token}(+\texttt{t}, \texttt{i}, \texttt{c}) \Rightarrow \texttt{InField}(\texttt{i}, +\texttt{f}, \texttt{c})$$

where all free variables are implicitly universally quantified. The "`+t`, `+f`" notation signifies that the MLN contains an instance of this rule for each *(token, field)* pair. If this rule was learned in isolation, the weight of the (t, f)th instance would be $\log(p_{tf}/(1 - p_{tf}))$, where p_{tf} is the corresponding entry in the HMM observation matrix.

In general, the transition matrix of the HMM is represented by a rule of the form

$$\texttt{InField}(\texttt{i}, +\texttt{f}, \texttt{c}) \Rightarrow \texttt{InField}(\texttt{i} + 1, +\texttt{f}', \texttt{c})$$

However, we (and others, e.g., [37]) have found that for segmentation it suffices to capture the basic regularity that consecutive positions tend to be part of the same field. Thus we replace $\texttt{f}'$ by $\texttt{f}$ in the formula above. We also impose the condition that a position in a citation string can be part of at most one field; it may be part of none.

The main shortcoming of this model is that it has difficulty pinpointing field boundaries. Detecting these is key for information extraction, and a number of approaches use rules designed specifically for this purpose (e.g., [60]). In citation matching, boundaries are usually marked by punctuation symbols. This can be incorporated into the MLN by modifying the rule above to

$$\texttt{InField}(\texttt{i}, +\texttt{f}, \texttt{c}) \wedge \neg\texttt{HasPunc}(\texttt{c}, \texttt{i}) \Rightarrow \texttt{InField}(\texttt{i} + 1, +\texttt{f}, \texttt{c})$$

The $\neg\texttt{HasPunc}(\texttt{c}, \texttt{i})$ precondition prevents propagation of fields across punctuation marks. Because propagation can occur differentially to the left and right, the MLN also contains the reverse form of the rule. In addition, to account for commas being weaker separators than other punctuation, the MLN includes versions of these rules with `HasComma`() instead of `HasPunc`().

Finally, the MLN contains the following rules[2]: the first two positions of a citation are usually in the author field, and the middle one in the title; initials (e.g., "J.") tend to

[2]The complete set of rules can be found in alchemy.cs.washington.edu/papers/poon07.

appear in either the author or the venue field; positions preceding the last non-venue initial are usually not part of the title or venue; and positions after the first venue keyword (e.g., "Proceedings", "Journal") are usually not part of the author or title. Despite its simplicity, this model is quite accurate, and forms a very competitive baseline for joint inference, as we will see in the experimental section.

3.3.2 Entity Resolution

As our starting point for entity resolution, we took the MLN of [99], which assumes pre-segmented citations. It contains rules of the form: if two fields contain many common tokens, they are the same; if the fields of two citations match, the citations also match, and vice-versa; etc. Simply taking the output `InField`() predicates of the segmentation MLN as evidence to this MLN would constitute a standard pipeline model. Merging the two MLNs produces a joint model for segmentation and entity resolution. We found, however, that this gives poor results, because entity resolution often leads segmentation astray. Since only a small fraction of citation pairs $(\mathtt{c}, \mathtt{c}')$ match, in the absence of strong evidence to the contrary the MLN will conclude that `SameCitation`$(\mathtt{c}, \mathtt{c}')$ is false. If `SameCitation`$(\mathtt{c}, \mathtt{c}')$ is the consequent of a rule (or rule chain) with `InField`() in the antecedent, the MLN may then infer that `InField`() is false, even if segmentation alone would correctly predict it to be true. This is an example of how joint inference can hurt accuracy.

Our solution to this problem is to define predicates and rules specifically for passing information between the stages, as opposed to just using the existing `InField`() outputs. We want a "higher bandwidth" of communication between segmentation and entity resolution, without letting excessive segmentation noise through. We accomplish this by defining a `SimilarTitle`$(\mathtt{c}, \mathtt{i}, \mathtt{j}, \mathtt{c}', \mathtt{i}', \mathtt{j}')$ predicate, which is true if citations $\mathtt{c}$ and $\mathtt{c}'$ contain similar title-like strings at positions $\mathtt{i}$ to $\mathtt{j}$ and $\mathtt{i}'$ to $\mathtt{j}'$, respectively. A string is title-like if it does not contain punctuation and does not match the "title exclusion" rules in the previous section. Two such strings are considered similar if they start with the same trigram and end with the same token. Most importantly, `SimilarTitle`() is true only if at least one of the strings is immediately preceded by punctuation, and at least one is immediately followed by punctuation. This greatly reduces the number of potential matches, focusing them on the cases that are most likely relevant for joint inference. In sum, $\texttt{SimilarTitle}()$ incorporates lower-level segmentation information into entity resolution.

If two citations have similar titles, they are usually the same, unless they appear in different venues (in which case they are usually different versions of the same paper). Hence the basic rule we use to perform entity resolution is

$$\texttt{SimilarTitle}(\texttt{c},\texttt{i},\texttt{j},\texttt{c}',\texttt{i}',\texttt{j}') \wedge \texttt{SimilarVenue}(\texttt{c},\texttt{c}') \Rightarrow \texttt{SameCitation}(\texttt{c},\texttt{c}')$$

coupled with the unit clause $\neg\texttt{SameCitation}(\texttt{c},\texttt{c}')$, which represents the default. $\texttt{SimilarVenue}(\texttt{c},\texttt{c}')$ is true as long as $\texttt{c}$ and $\texttt{c}'$ do not contain conflicting venue keywords (e.g., "Journal" in one and "Proceedings" in the other). The rule above ignores whether the citations' authors are the same, which may seem unintuitive. However, the reason is simple. When two citations have the same title and venue, they almost always have the same author as well, and thus comparing authors contributes no additional information. On the other hand, if the authors are the same but are difficult to match, as is often the case, including $\texttt{SimilarAuthor}(\texttt{c},\texttt{c}')$ as a precondition in the rule above prevents drawing the correct conclusion that the citations are the same. Thus, on balance, including $\texttt{SimilarAuthor}(\texttt{c},\texttt{c}')$ is detrimental.

As we will see in the experimental section, this simple MLN suffices to outperform the state of the art in citation matching.

3.3.3 Joint Segmentation

Segmenting a citation can help segment similar ones. For example, if in one citation the title is clearly delimited by punctuation, but in a similar one it is not, noticing this similarity can help extract the more difficult title. Incorporating this idea into the MLN described earlier leads to *joint segmentation*, where citations are segmented collectively, as opposed to in isolation. As before, we proceed by defining a predicate for this purpose. $\texttt{JointInferenceCandidate}(\texttt{c},\texttt{i},\texttt{c}')$ is true if the trigram starting at position $\texttt{i}$ in citation $\texttt{c}$ also appears somewhere in citation $\texttt{c}'$, the trigrams do not match the "title exclusion" rules, and the trigram in $\mathtt{c}$ is not preceded by punctuation, while in $\mathtt{c}'$ it is. This rule thus identifies potential opportunities for one title segmentation to help another. We then incorporate this into the segmentation model simply by adding a precondition to the "field propagation" rules. For example:

$$\begin{gathered}\mathtt{InField(i,+f,c)} \wedge \neg\mathtt{HasPunc(c,i)}\\ \wedge(\neg\exists\mathtt{c}'\ \mathtt{JointInferenceCandidate(c,i,c')})\\ \Rightarrow \mathtt{InField(i+1,+f,c)}\end{gathered}$$

The effect of this precondition is to potentially introduce a field boundary in $\mathtt{c}$ immediately before a substring if a similar title-like substring is preceded by punctuation in another citation. This may be quite effective when citation lists are "sparse" (i.e., strings usually do not occur both at boundaries and inside fields). However, if the citation lists are "dense," it may produce incorrect field boundaries. Consider the following two citations from the Cora dataset:

R. Schapire. On the strength of weak learnability. Proceedings of the 30th I.E.E.E. Symposium on the Foundations of Computer Science, 1989, pp. 28-33.

Robert E. Schapire. 5(2) The strength of weak learnability. Machine Learning, 1990 197-227,

In the second citation, "The strength of" is immediately preceded by punctuation, which will cause a title boundary to be incorrectly introduced between "On" and "the" in the first citation. To combat this, we can in addition require that the two citations in fact resolve to the same entity:

$$\begin{gathered}\mathtt{InField(i,+f,c)} \wedge \neg\mathtt{HasPunc(c,i)}\\ \wedge(\neg\exists\mathtt{c}'\ \mathtt{JointInferenceCandidate(c,i,c')} \wedge \mathtt{SameCitation(c,c')})\\ \Rightarrow \mathtt{InField(i+1,+f,c)}\end{gathered}$$

This is another instance of joint inference between segmentation and entity resolution: in this case, entity resolution helps perform segmentation. However, this rule may sometimes be too strict; consider the following citations from CiteSeer:

H. Maruyama Constraint dependency grammar and its weak generative capacity. Computer Software, 1990.

Maruyama, H 1990. Constraint dependency grammar. Technical Report # RT 0044, IBM, Tokyo, Japan.

In this case, the first rule would correctly place a title boundary before "Constraint", while the second one would not. Both approaches are generally applicable and, as shown in the next section, significantly outperform isolated segmentation. The first one (which we will call Jnt-Seg) is better for sparse datasets (like CiteSeer), and the second (Jnt-Seg-ER) for dense ones (like Cora).

3.4 Inference and Learning

We used MC-SAT for inference and modified the standard algorithm for supervised weight learning (see Section 2.5) in two ways. First, we used MC-SAT to compute the expected counts when estimating the gradient. MC-SAT is much more efficient than Gibbs sampling, and tends to give more reliable results than MaxWalkSAT since the distribution can contain many modes. Second, we used a different learning rate for each weight, by dividing the global learning rate by the number of true groundings of the corresponding clause. This greatly speeds up learning. Because the number of true groundings can vary widely, a learning rate that is small enough to avoid divergence in some weights is too small for fast convergence in others. Having separate learning rates overcomes this problem. Specifically, our adjustment can be viewed as preconditioning using the inverse diagonal Hessian,[3] with the variances approximated by the number of true groundings. We also modified Alchemy to save time

[3]This achieves the optimal learning rates if the formulas are independent from each other (e.g., as in logistic regression). See [68] for more details.

Table 3.1: CiteSeer entity resolution: cluster recall on each section.

Approach	Constr.	Face	Reason.	Reinfor.
Fellegi-Sunter	84.3	81.4	71.3	50.6
Lawrence (1999)	89	94	86	79
Pasula (2003)	93	97	96	94
Wellner (2004)	95.1	96.9	93.7	94.7
Joint MLN	96.0	97.1	95.1	96.7

Table 3.2: Cora entity resolution: pairwise recall/precision and cluster recall.

Approach	Pairwise Rec./Prec.	Cluster Recall
Fellegi-Sunter	78.0 / 97.7	62.7
Joint MLN	94.3 / 97.0	78.1

and memory in grounding the network.[4]

3.5 Experiments

3.5.1 Datasets

We experimented on CiteSeer and Cora, which are standard datasets for citation matching. The CiteSeer dataset was created by [61] and used in [82] and [114]. The Cora dataset was created by Andrew McCallum and later segmented by [9]. We fixed a number of errors in the labels.[5]

In citation matching, a *cluster* is a set of citations that refer to the same paper, and a

[4]By default, Alchemy converts an existentially quantified formula into a disjunction by enumerating the values of the quantified variables. We simplified this disjunction by skipping the components that are guaranteed to be false given the evidence.

[5]Dataset available at http://alchemy.cs.washington.edu/papers/poon07.

Table 3.3: CiteSeer segmentation: F1 on each section.

All	Total	Author	Title	Venue
Isolated	94.4	94.6	92.8	95.3
Jnt-Seg	94.5	94.9	93.0	95.3
Jnt-Seg-ER	94.5	94.8	93.0	95.3
Nontrivial	Total	Author	Title	Venue
Isolated	94.4	94.5	92.5	95.5
Jnt-Seg	94.9	95.2	93.6	95.7
Jnt-Seg-ER	94.8	95.1	93.3	95.6
Potential	Total	Author	Title	Venue
Isolated	91.7	91.5	86.5	94.9
Jnt-Seg	94.3	94.5	92.4	95.3
Jnt-Seg-ER	93.9	94.4	92.0	94.9

nontrivial cluster contains more than one citation. The CiteSeer dataset has 1563 citations and 906 clusters. It consists of four sections, each on a different topic. Over two-thirds of the clusters are singletons, and the largest contains 21 citations. The Cora dataset has 1295 citations and 134 clusters. Unlike in CiteSeer, almost every citation in Cora belongs to a nontrivial cluster; the largest cluster contains 54 citations.

3.5.2 Methodology

For CiteSeer, we followed [114] and used the four sections for four-fold cross-validation. In Cora, we randomly split citations into three subsets so that they were distributed as evenly as possible and no cluster was broken apart (to avoid train-test contamination). We then used these subsets for three-fold cross-validation. In CiteSeer, the largest section yields 0.3 million query atoms and 0.4 million ground clauses; in Cora, 0.26 million and 0.38 million, respectively. For simplicity, we took the original author and title fields and combined other

Table 3.4: Cora segmentation: F1 on each section.

All	Total	Author	Title	Venue
Isolated	98.2	99.3	97.3	98.2
Jnt-Seg	98.4	99.5	97.5	98.3
Jnt-Seg-ER	98.4	99.5	97.6	98.3
Nontrivial	Total	Author	Title	Venue
Isolated	98.3	99.4	97.4	98.2
Jnt-Seg	98.5	99.5	97.7	98.3
Jnt-Seg-ER	98.5	99.5	97.7	98.3
Potential	Total	Author	Title	Venue
Isolated	97.6	97.9	95.2	98.8
Jnt-Seg	98.7	99.3	97.9	99.0
Jnt-Seg-ER	98.8	99.3	97.9	99.0

fields such as booktitle, journal, pages, and publishers into venue.

In preprocessing, we carried out standard normalizations like conversion to lower case, simple stemming, and hyphen removal. For token matching, we used edit distance and concatenation. Two tokens match if their edit distance is at most one, and a bigram matches the unigram formed by concatenating the two tokens.

In MLN weight learning, we used 30 iterations of gradient descent, and in each iteration, for each fold, we ran MC-SAT for 20 seconds. This proves to be effective and enables fast learning; see [42] for a theoretical justification of a similar method. However, it also introduces fluctuations among the runs, and we obviate these by reporting averages of ten runs. The total learning time per run ranged from 20 minutes to an hour. In inference, we used no burn-in and let MC-SAT run for 30 minutes. MC-SAT returns marginal probabilities of query atoms. We predict an atom is true if its probability is at least 0.5; otherwise we predict false.

Table 3.5: Segmentation: wins and losses in F1.

CiteSeer	JS-Isolated	JSR-Isolated	JSR-JS
All	8-2	4-2	1-7
Nontrivial	8-2	9-2	1-8
Potential	12-0	9-3	2-9
Cora	JS-Isolated	JSR-Isolated	JSR-JS
All	5-1	6-1	3-1
Nontrivial	4-1	5-1	2-1
Potential	7-1	8-1	5-0

For entity resolution in CiteSeer, we measured *cluster recall* for comparison with previously published results. Cluster recall is the fraction of clusters that are correctly output by the system after taking transitive closure from pairwise decisions. For entity resolution in Cora, we measured both cluster recall and pairwise recall/precision. In both datasets we also compared with a "standard" Fellegi-Sunter model (see [99]), learned using logistic regression, and with oracle segmentation as the input. For segmentation, we measured F1 for `InField`, F1 being the harmonic mean of recall and precision.

3.5.3 Results

Table 3.1 shows that our approach outperforms previous ones on CiteSeer entity resolution. (Results for Lawrence (1999), Pasula (2003) and Wellner (2004) are taken from the corresponding papers. The MLN results are for Jnt-Seg; the results for Jnt-Seg-ER are identical, except for 95.5 instead of 96.0 on Constr.) This is particularly notable given that the models of [82] and [114] involved considerably more knowledge engineering than ours, contained more learnable parameters, and used additional training data.

Table 3.2 shows that our entity resolution approach easily outperforms Fellegi-Sunter on Cora, and has very high pairwise recall/precision. (As before, the MLN results are for Jnt-

Table 3.6: Percentage of F1 error reduction in segmentation obtained by joint inference.

CiteSeer	Total	Author	Title	Venue
All	2	6	3	0
Nontrivial	9	13	15	4
Potential	31	35	44	8
Cora	Total	Author	Title	Venue
All	11	24	7	6
Nontrivial	12	17	12	6
Potential	50	67	56	17

Seg; the results for Jnt-Seg-ER are identical, except for 75.2 instead of 78.1 on cluster recall. Note that in Cora clusters are much more difficult to discriminate than in CiteSeer; also, because Cora contains on average only 44 clusters per fold, each less-than-perfect cluster degrades cluster recall by over 2%.)

Table 3.3 and Table 3.4 compare isolated segmentation with the two joint inference methods on CiteSeer and Cora. The "Total" column aggregates results on all `InField` atoms, and the remaining columns show results for each field type. The "All" section shows results for all citations, "Nontrivial" for non-singleton citations, and "Potential" for citations with poor author-title boundary (and which therefore might benefit from joint inference). Approximately 9% of citations fall into the "Potential" category in CiteSeer, and 21% in Cora. All total improvements are statistically significant at the 1% level using McNemar's test. Table 3.5 summarizes the relative performance of the three methods in terms of number of wins and losses. Each *(field, subset)* pair is one trial. In CiteSeer, there are four subsets and twelve comparisons, and thus 12-0 means that the first method outperformed the second in all comparisons. In Cora, there are three subsets and nine comparisons in total, so 5-0 means the first wins five times and ties four times.

Joint inference consistently outperforms isolated inference. Table 3.6 shows error reduction from isolated inference by the better-performing joint inference method (Jnt-Seg in CiteSeer, Jnt-Seg-ER in Cora). Improvement among potential citations is substantial: in Cora, error is reduced by half in total, 67% for authors, and 56% for titles; in CiteSeer, error is reduced by 31% in total, 35% for authors, and 44% for titles. Overall in Cora, joint inference reduces error by 11% in total and 24% for authors. The overall improvement in CiteSeer is not as large, because the percentage of potential citations is much smaller. The joint inference methods do not impact performance on citations with good boundaries (not shown separately in these tables).

In CiteSeer, Jnt-Seg outperforms Jnt-Seg-ER, while in Cora the opposite is true. These results are consistent with the different characteristics of the two datasets: Cora is dense, while CiteSeer is sparse. In Cora, there is a lot of word overlap among citations, and often the same trigram occurs both within a field and at the boundary, making Jnt-Seg more prone to misaligning boundaries. In CiteSeer, however, this rarely happens and Jnt-Seg-ER, being conservative, can miss opportunities to improve segmentation. There is no absolute advantage of one joint inference method over the other; the best choice depends on the dataset.

3.6 Discussion

To overcome the difficulty in joint inference, our solution introduced special predicates and formulas that were tailored to the citation domain for passing information between processing stages. With Markov logic, the engineering effort takes place mostly at the formula level, which is considerably easier compared to engineering at the software and algorithmic level. However, it still requires considerable amount of effort and makes it harder to generalize to other domains. Ideally, we would like to have a modular approach that works by simply combining the component MLNs. This has proven to give poor results in our preliminary experiments, and the reason is likely due to difficulties in approximate inference. Inference in the combined MLNs involved longer chains of reasoning and is considerably more challenging for the underlying SAT solvers. Effectively, our solution alleviates the inference complexity by introducing auxiliary predicates to propagate information. Therefore, a key research direction is to improve the accuracy and efficiency of inference algorithms, so that such domain-specific engineering can be further alleviated.

3.7 Summary

Information extraction is an important form of machine reading that focuses on extracting database records according to a predefined schema. Joint inference can potentially improve accuracy by propagating information across pipeline stages as well as among objects. However, joint inference is often difficult to perform effectively, due to its complexity, computational cost, and sometimes mixed effect on accuracy. In this chapter, we show how these problems can be addressed in a citation matching domain, using Markov logic, the MC-SAT algorithm, and careful interconnection of inference tasks. Experiments on standard datasets show that our approach is efficient and consistently improves accuracy over non-joint inference. It is also more accurate and easier to set up than previous semi-joint approaches.

Chapter 4
JOINT UNSUPERVISED COREFERENCE RESOLUTION

4.1 Introduction

For machine reading in free text, an important subtask is *coreference resolution*, which identifies *mentions* (typically noun phrases) that refer to the same *entities*. Machine learning approaches to coreference resolution are typically supervised, which treat the problem as one of classification: for each pair of mentions, predict whether they corefer or not (e.g., [73]). While successful, these approaches require labeled training data, consisting of mention pairs and the correct decisions for them. This limits their applicability. Unsupervised approaches are attractive due to the availability of large quantities of unlabeled text. However, unsupervised coreference resolution is much more difficult. Haghighi and Klein's (2007) model [39], the most sophisticated before this work, still lags supervised ones by a substantial margin. Extending it appears difficult, due to the limitations of its Dirichlet process-based representation.

The lack of label information in unsupervised coreference resolution can potentially be overcome by performing joint inference, which leverages the "easy" decisions to help make related "hard" ones. Relations that have been exploited in supervised coreference resolution include transitivity [73] and anaphoricity [22]. However, there is little work to date on joint inference for unsupervised coreference resolution.

In this chapter, we develop the first unsupervised approach that is competitive with supervised ones. This is made possible by performing joint inference across mentions, in contrast to the pairwise classification typically used in supervised methods, and by using Markov logic as a representation language, which enables us to easily express relations like apposition and predicate nominals. By extending the state-of-the-art algorithms for inference and learning, we developed the first general-purpose unsupervised learning algorithm for Markov logic, and applied it to unsupervised coreference resolution.

We test our approach on standard MUC and ACE datasets. Our basic model, trained on a minimum of data, suffices to outperform Haghighi and Klein's (2007) one. Our full model, using apposition and other relations for joint inference, is often as accurate as the best supervised models, or more.

4.2 Related Work

Most existing supervised learning approaches for coreference resolution are suboptimal since they resolve each mention pair independently, only imposing transitivity in postprocessing [78]. Moreover, many of them break up the resolution step into subtasks (e.g., first determine whether a mention is anaphoric, then classify whether it is coreferent with an antecedent), which further forsakes opportunities for joint inference that have been shown to be helpful [88]. Using graph partitioning, McCallum & Wellner (2005) incorporated transitivity into pairwise classification and achieved the state-of-the-art result on the MUC-6 dataset [73], but their approach can only leverage one binary relation at a time, not arbitrary relations among mentions. Denis & Baldridge (2007) determined anaphoricity and pairwise classification jointly using integer programming [22], but they did not incorporate transitivity or other relations.

While potentially more appealing, unsupervised learning is very challenging, and unsupervised coreference resolution systems are still rare to this date. Prior to our work, the best performance in unsupervised coreference resolution was achieved by Haghighi & Klein (2007), using a nonparametric Bayesian model based on hierarchical Dirichlet processes [39]. At the heart of their system is a mixture model with a few linguistically motivated features such as head words, entity properties and salience. Their approach is a major step forward in unsupervised coreference resolution, but extending it is challenging. The main advantage of Dirichlet processes is that they are exchangeable, allowing parameters to be integrated out, but Haghighi and Klein forgo this when they introduce salience. Their model thus requires Gibbs sampling over both assignments and parameters, which can be very expensive. Haghighi and Klein circumvent this by making approximations that potentially hurt accuracy. At the same time, the Dirichlet process prior favors skewed cluster sizes and a number of clusters that grows logarithmically with the number of data points, neither of which seems generally appropriate for coreference resolution.

Further, deterministic or strong non-deterministic dependencies cause Gibbs sampling to break down [87], making it difficult to leverage many linguistic regularities. For example, apposition (as in “Bill Gates, the chairman of Microsoft”) suggests coreference, and thus the two mentions it relates should always be placed in the same cluster. However, Gibbs sampling can only move one mention at a time from one cluster to another, and this is unlikely to happen, because it would require breaking the apposition rule. Blocked sampling can alleviate this problem by sampling multiple mentions together, but it requires that the block size be predetermined to a small fixed number. When we incorporate apposition and other regularities the blocks can become arbitrarily large, making this infeasible. For example, suppose we also want to leverage predicate nominals (i.e., the subject and the predicating noun of a copular verb are likely coreferent). Then a sentence like “He is Bill Gates, the chairman of Microsoft” requires a block of four mentions: “He”, “Bill Gates”, “the chairman of Microsoft”, and “Bill Gates, the chairman of Microsoft”. Similar difficulties occur with other inference methods. Thus, extending Haghighi and Klein’s model to include richer linguistic features is a challenging problem.

Our approach is instead based on Markov logic, a powerful representation for joint inference with uncertainty [23]. Like Haghighi and Klein’s, our model is cluster-based rather than pairwise, and implicitly imposes transitivity. We do not predetermine anaphoricity of a mention, but rather fuse it into the integrated resolution process. As a result, our model is inherently joint among mentions and subtasks like determining anaphoricity or coreference. It shares several features with Haghighi & Klein’s model, but removes or refines features where we believe it is appropriate to. Most importantly, our model leverages apposition and predicate nominals, which Haghighi & Klein did not use. We show that this can be done very easily in our framework, and yet results in very substantial accuracy gains.

It is worth noticing that Markov logic is also well suited for joint inference in supervised systems (e.g., transitivity, which took McCallum & Wellner (2005) nontrivial effort to incorporate, can be handled in Markov logic with the addition of a single formula [89]).

4.3 An MLN for Joint Unsupervised Coreference Resolution

In this section, we present our MLN for joint unsupervised coreference resolution. Our model deviates from Haghighi & Klein's (2007) in several important ways. First, our MLN does not model saliences for proper nouns or nominals, as their influence is marginal compared to other features; for pronoun salience, it uses a more intuitive and simpler definition based on distance, and incorporated it as a prior. Another difference is in identifying heads. For the ACE datasets, Haghighi and Klein used the gold heads; for the MUC-6 dataset, where labels are not available, they crudely picked the rightmost token in a mention. We show that a better way is to determine the heads using head rules in a parser. This improves resolution accuracy and is always applicable. Crucially, our MLN leverages syntactic relations such as apposition and predicate nominals, which are not used by Haghighi and Klein. In our approach, what it takes is just adding two formulas to the MLN.

As common in previous work, we assume that true mention boundaries are given. We do not assume any other labeled information. In particular, we do not assume gold name entity recognition (NER) labels, and unlike Haghighi & Klein (2007), we do not assume gold mention types (for ACE datasets, they also used gold head words). We determined the head of a mention either by taking its rightmost token, or by using the head rules in a parser. We detected pronouns using a list.

4.3.1 Base MLN

The main query predicate is $\texttt{InClust}(\texttt{m}, \texttt{c}!)$, which is true iff mention $\texttt{m}$ is in cluster $\texttt{c}$. The "$\texttt{c}!$" notation signifies that for each $\texttt{m}$, this predicate is true for a unique value of $\texttt{c}$. The main evidence predicate is $\texttt{Head}(\texttt{m}, \texttt{t}!)$, where $\texttt{m}$ is a mention and $\texttt{t}$ a token, and which is true iff $\texttt{t}$ is the head of $\texttt{m}$. A key component in our MLN is a simple head mixture model, where the mixture component priors are represented by the unit clause

$$\texttt{InClust}(+\texttt{m}, +\texttt{c})$$

and the head distribution is represented by the *head prediction rule*

$$\texttt{InClust}(\texttt{m}, +\texttt{c}) \wedge \texttt{Head}(\texttt{m}, +\texttt{t}).$$

All free variables are implicitly universally quantified. The "+" notation signifies that the MLN contains an instance of the rule, with a separate weight, for each value combination of the variables with a plus sign.

By convention, at each inference step we name each non-empty cluster after the earliest mention it contains. This helps break the symmetry among mentions, which otherwise produces multiple optima and makes learning unnecessarily harder. To encourage clustering, we impose an exponential prior on the number of non-empty clusters with weight -1. (This is incorporated into the MLN as a formula with fixed weight.)

The above model only clusters mentions with the same head, and does not work well for pronouns. To address this, we introduce the predicate $\texttt{IsPrn(m)}$, which is true iff the mention $\texttt{m}$ is a pronoun, and adapt the head prediction rule as follows:

$$\neg\texttt{IsPrn}(\texttt{m}) \wedge \texttt{InClust}(\texttt{m}, +\texttt{c}) \wedge \texttt{Head}(\texttt{m}, +\texttt{t})$$

This is always false when $\texttt{m}$ is a pronoun, and thus applies only to non-pronouns.

Pronouns tend to resolve with mentions that are semantically compatible with them. Thus we introduce predicates that represent entity type, number, and gender: $\texttt{Type}(\texttt{x}, \texttt{e}!)$, $\texttt{Number}(\texttt{x}, \texttt{n}!)$, $\texttt{Gender}(\texttt{x}, \texttt{g}!)$, where $\texttt{x}$ can be either a cluster or mention,
$\texttt{e} \in \{\texttt{Person}, \texttt{Organization}, \texttt{Location}, \texttt{Other}\}$, $\texttt{n} \in \{\texttt{Singular}, \texttt{Plural}\}$ and
$\texttt{g} \in \{\texttt{Male}, \texttt{Female}, \texttt{Neuter}\}$. Many of these are known for pronouns, and some can be inferred from simple linguistic cues (e.g., "Ms. Galen" is a singular female person, while "XYZ Corp." is an organization).[1] Entity type assignment is represented by the unit clause

$$\texttt{Type}(+\texttt{x}, +\texttt{e})$$

and similarly for number and gender. A mention should agree with its cluster in entity type. This is ensured by the hard rule (which has infinite weight and must be satisfied)

$$\texttt{InClust}(\texttt{m}, \texttt{c}) \Rightarrow (\texttt{Type}(\texttt{m}, \texttt{e}) \Leftrightarrow \texttt{Type}(\texttt{c}, \texttt{e}))$$

There are similar hard rules for number and gender.

[1]We used the following cues: Mr., Ms., Jr., Inc., Corp., corporation, company. The proportions of known properties range from 14% to 26%.

Different pronouns prefer different entity types, as represented by

$$\texttt{IsPrn(m)} \wedge \texttt{InClust(m, c)} \wedge \texttt{Head(m, +t)} \wedge \texttt{Type(c, +e)}$$

which only applies to pronouns, and whose weight is positive if pronoun `t` is likely to assume entity type `e` and negative otherwise. There are similar rules for number and gender.

Aside from semantic compatibility, pronouns tend to resolve with nearby mentions. To model this, we impose an exponential prior on the distance (number of mentions) between a pronoun and its antecedent, with weight -1.[2] This is similar to Haghighi and Klein's treatment of salience, but simpler.

4.3.2 Full MLN

Syntactic relations among mentions often suggest coreference. Incorporating such relations into our MLN is straightforward. We illustrate this with two examples: apposition and predicate nominals. We introduce a predicate for apposition, `Appo(x, y)`, where `x, y` are mentions, and which is true iff `y` is an appositive of `x`. We then add the rule

$$\texttt{Appo(x, y)} \Rightarrow (\texttt{InClust(x, c)} \Leftrightarrow \texttt{InClust(y, c)})$$

which ensures that x, y are in the same cluster if y is an appositive of x. Similarly, we introduce a predicate for predicate nominals, `PredNom(x, y)`, and the corresponding rule.[3] The weights of both rules can be learned from data with a positive prior mean. For simplicity, in this paper we treat them as hard constraints.

4.3.3 Rule-Based MLN

We also consider a rule-based system that clusters non-pronouns by their heads, and attaches a pronoun to the cluster which has no known conflicting type, number, or gender, and

[2]For simplicity, if a pronoun has no antecedent, we define the distance to be ∞. So a pronoun must have an antecedent in our model, unless it is the first mention in the document or it can not resolve with previous mentions without violating hard constraints. It is straightforward to soften this with a finite penalty.

[3]We detected apposition and predicate nominatives using two simple heuristics based on parses: if (NP, comma, NP) are the first three children of an NP, then any two of the three noun phrases are apposition; if (NP NP) form a noun-noun compound, then the two noun phrases are apposition.

contains the closest antecedent for the pronoun. This system can be encoded in an MLN with just four rules. Three of them are the ones for enforcing agreement in type, number, and gender between a cluster and its members, as defined in the base MLN. The fourth rule is

$$\begin{gathered}\neg\texttt{IsPrn(m1)} \wedge \neg\texttt{IsPrn(m2)} \\ \wedge\texttt{Head(m1,h1)} \wedge \texttt{Head(m2,h2)} \\ \wedge\texttt{InClust(m1,c1)} \wedge \texttt{InClust(m2,c2)} \\ \Rightarrow (\texttt{c1} = \texttt{c2} \Leftrightarrow \texttt{h1} = \texttt{h2}).\end{gathered}$$

With a large but not infinite weight (e.g., 100), this rule has the effect of clustering non-pronouns by their heads, except when it violates the hard rules. The MLN can also include the apposition and predicate-nominal rules. As in the base MLN, we impose the same exponential prior on the number of non-empty clusters and that on the distance between a pronoun and its antecedent. This simple MLN is remarkably competitive, as we will see in the experiment section.

4.4 Inference and Learning

Unsupervised learning in Markov logic maximizes the conditional log-likelihood

$$L(x, y) = \log P(Y = y|X = x) = \log \sum_z P(Y = y, Z = z|X = x)$$

where Z are unknown predicates. In our coreference resolution MLN, Y includes `Head` and known groundings of `Type`, `Number` and `Gender`, Z includes `InClust` and unknown groundings of `Type`, `Number`, `Gender`, and X includes `IsPrn`, `Appo` and `PredNom`. (For simplicity, from now on we drop X from the formula.) With Z, the optimization problem is no longer convex. However, we can still find a local optimum using gradient descent, with the gradient being

$$\frac{\partial}{\partial w_i} L(y) = E_{Z|y}[n_i] - E_{Y,Z}[n_i]$$

where n_i is the number of true groundings of the ith clause. Compared to supervised learning, the key difference is that now the gradient is the difference of two expectations. The first expectation ranges over arbitrary Z while fixing Y to the training data, whereas the second expectation ranges over arbitrary Y and Z.

Note that if some labeled examples are available, semi-supervised learning can be done straightforwardly by fixing Z to the known labels in the first expectation [86].

The gradient can be computed by approximating both expectations using MC-SAT. To combat the ill-conditioned problem, we extended the preconditioned scaled conjugate gradient algorithm (PSCG) that was previously developed for supervised weight learning [68]. In PSCG, the optimal step size in each step is estimated as

$$\alpha = \frac{-d^T g}{d^T H d + \lambda d^T d}.$$

where g is the gradient, d the conjugate update direction, and λ a parameter that is automatically tuned to trade off second-order information with gradient descent. H is the Hessian matrix, with the (i,j)th entry being

$$\frac{\partial^2}{\partial w_i \partial w_j} L(y) = E_Y[n_i] \cdot E_Y[n_j] - E_Y[n_i \cdot n_j] = -Cov_Y[n_i, n_j].$$

The Hessian can be approximated with the same samples used for the gradient. Its negative inverse diagonal is used as the preconditioner.[4]

In unsupervised learning, the (i,j)th entry of the Hessian now becomes

$$\frac{\partial^2}{\partial w_i \partial w_j} L(y) = Cov_{Z|y}[n_i, n_j] - Cov_{Y,Z}[n_i, n_j]$$

and the step size can be computed accordingly. Since our problem is no longer convex, the negative diagonal Hessian may contain zero or negative entries, so we first took the absolute values of the diagonal and added 1, then used the inverse as the preconditioner. For automatically adjusting λ, Lowd & Domingos (2007) used a more aggressive schedule than the one suggested in [81] by leveraging the convexity of the learning objective. In unsupervised learning, however, we found that this can lead to divergence since the learning objective is no longer convex, so we roll back to the schedule in [81].

Notice that when the objects form independent subsets (in our cases, mentions in each document), we can process them in parallel and then gather sufficient statistics for learning. We developed an efficient parallelized implementation of our unsupervised learning

[4]Lowd & Domingos showed that α can be computed more efficiently, without explicitly approximating or storing the Hessian. Readers are referred to their paper for details.

algorithm using the message-passing interface (MPI). Learning in MUC-6 took only one hour, and in ACE-2004 two and a half.

To reduce burn-in time, we initialized MC-SAT with the state returned by MaxWalkSAT [44], rather than a random solution to the hard clauses. In the existing implementation in Alchemy [57], SampleSAT flips only one atom in each step, which is inefficient for predicates with unique-value constraints (e.g., $\texttt{Head}(\texttt{m}, \texttt{c!})$). Such predicates can be viewed as multi-valued predicates (e.g., $\texttt{Head}(\texttt{m})$ with value ranging over all $\texttt{c}$'s) and are prevalent in NLP applications. We adapted SampleSAT to flip two or more atoms in each step so that the unique-value constraints are automatically satisfied. By default, MC-SAT treats each ground clause as a separate factor while determining the slice. This can be very inefficient for highly correlated clauses. For example, given a non-pronoun mention $\texttt{m}$ currently in cluster $\texttt{c}$ and with head $\texttt{t}$, among the mixture prior rules involving $\texttt{m}$ $\texttt{InClust}(\texttt{m}, \texttt{c})$ is the only one that is satisfied, and among those head-prediction rules involving $\texttt{m}$, $\neg\texttt{IsPrn}(\texttt{m}) \wedge \texttt{InClust}(\texttt{m}, \texttt{c}) \wedge \texttt{Head}(\texttt{m}, \texttt{t})$ is the only one that is satisfied; the factors for these rules multiply to $\phi = \exp(w_{\texttt{m},\texttt{c}} + w_{\texttt{m},\texttt{c},\texttt{t}})$, where $w_{\texttt{m},\texttt{c}}$ is the weight for $\texttt{InClust}(\texttt{m}, \texttt{c})$, and $w_{\texttt{m},\texttt{c},\texttt{t}}$ is the weight for $\neg\texttt{IsPrn}(\texttt{m}) \wedge \texttt{InClust}(\texttt{m}, \texttt{c}) \wedge \texttt{Head}(\texttt{m}, \texttt{t})$, since an unsatisfied rule contributes a factor of $e^0 = 1$. We extended MC-SAT to treat each set of mutually exclusive and exhaustive rules as a single factor. E.g., for the above $\texttt{m}$, MC-SAT now samples u uniformly from $(0, \phi)$, and requires that in the next state ϕ' be no less than u. Equivalently, the new cluster and head for $\texttt{m}$ should satisfy $w_{\texttt{m},\texttt{c}'} + w_{\texttt{m},\texttt{c}',\texttt{t}'} \geq \log(u)$. We extended SampleSAT so that when it considers flipping any variable involved in such constraints (e.g., $\texttt{c}$ or $\texttt{t}$ above), it ensures that their new values still satisfy these constraints.

The final clustering is found using the MaxWalkSAT weighted satisfiability solver [44], with the same extensions as mentioned above for SampleSAT. We first ran a MaxWalkSAT pass with only finite-weight formulas, then ran another pass with all formulas. We found that this significantly improved the quality of the results that MaxWalkSAT returned.

4.5 Experiments

4.5.1 System

We implemented our method as an extension to the Alchemy system [57]. Since our learning uses sampling, all results are the average of five runs using different random seeds. Our optimization problem is not convex, so initialization is important. The core of our model (head mixture) tends to cluster non-pronouns with the same head. Therefore, we initialized by setting all weights to zero, and running the same learning algorithm on the base MLN, while assuming that in the ground truth, non-pronouns are clustered by their heads. (Effectively, the corresponding `InClust` atoms are assigned to appropriate values and are included in Y rather than Z during learning.) We used 30 iterations of PSCG for learning. (In preliminary experiments, additional iterations had little effect on coreference accuracy.) We generated 100 samples using MC-SAT for each expectation approximation.[5]

4.5.2 Methodology

We conducted experiments on MUC-6, ACE-2004, and ACE Phrase-2 (ACE-2). We evaluated our systems using two commonly-used scoring programs: MUC [111] and B^3 [4]. To gain more insight, we also report pairwise resolution scores and mean absolute error in the number of clusters.

The MUC-6 dataset consists of 30 documents for testing and 221 for training. To evaluate the contribution of the major components in our model, we conducted five experiments, each differing from the previous one in a single aspect. We emphasize that our approach is unsupervised, and thus the data only contains raw text plus true mention boundaries.

MLN-1 In this experiment, the base MLN was used, and the head was chosen simply as the rightmost token in a mention. Our system was run on each test document separately, using a minimum of training data (the document itself).

[5]Each sample actually contains a large number of groundings, so 100 samples yield sufficiently accurate statistics for learning.

MLN-30 Our system was trained on all 30 test documents together. This tests how much can be gained by pooling information.

MLN-H The heads were determined using the head rules in the Stanford parser [50], plus simple heuristics to handle suffixes such as "Corp." and "Inc."

MLN-HA The apposition rule was added.

MLN-HAN The predicate-nominal rule was added. This is our full model.

We also compared with two rule-based MLNs: **RULE** chose the head simply as the rightmost token in a mention, and did not include the apposition rule and predicate-nominal rule; **RULE-HAN** chose the head using the head rules in the Stanford parser, and included the apposition rule and predicate-nominal rule.

Past results on ACE were obtained on different releases of the datasets, e.g., Haghighi and Klein (2007) used the ACE-2004 training corpus, Denis and Baldridge (2007) used ACE Phrase-2, and Culotta *et al.* (2007) used the ACE-2004 formal test set. In this paper, we used the ACE-2004 training corpus and ACE Phrase-2 (ACE-2) to enable direct comparisons with Haghighi & Klein (2007) and Denis and Baldridge (2007). Due to license restrictions, we were not able to obtain the ACE-2004 formal test set and so cannot compare directly to Culotta *et al.* (2007). The English version of the ACE-2004 training corpus contains two sections, BNEWS and NWIRE, with 220 and 128 documents, respectively. ACE-2 contains a training set and a test set. In our experiments, we only used the test set, which contains three sections, BNEWS, NWIRE, and NPAPER, with 51, 29, and 17 documents, respectively.

4.5.3 Results

Table 4.1 compares our system with previous approaches on the MUC-6 dataset, in MUC scores. Our approach greatly outperformed Haghighi & Klein (2007), the state-of-the-art unsupervised system. Our system, trained on individual documents, achieved an F1 score more than 7% higher than theirs trained on 60 documents, and still outperformed it trained

Table 4.1: Comparison of coreference results in MUC scores on the MUC-6 dataset.

	# Doc.	Prec.	Rec.	F1
H&K	60	80.8	52.8	63.9
H&K	381	80.4	62.4	70.3
M&W	221	-	-	73.4
RULE	-	76.0	65.9	70.5
RULE-HAN	-	81.3	72.7	76.7
MLN-1	1	76.5	66.4	71.1
MLN-30	30	77.5	67.3	72.0
MLN-H	30	81.8	70.1	75.5
MLN-HA	30	82.7	75.1	78.7
MLN-HAN	30	83.0	75.8	79.2

on 381 documents. Training on the 30 test documents together resulted in a significant gain. (We also ran experiments using more documents, and the results were similar.) Better head identification (MLN-H) led to a large improvement in accuracy, which is expected since for mentions with a right modifier, the rightmost tokens confuse rather than help coreference (e.g., "the chairman of Microsoft"). Notice that with this improvement our system already outperforms a state-of-the-art supervised system [73]. Leveraging apposition resulted in another large improvement, and predicate nominals also helped. Our full model scores about 9% higher than Haghighi & Klein (2007), and about 6% higher than McCallum & Wellner (2005). The B^3 scores of MLN-HAN on the MUC-6 dataset are 77.4 (precision), 67.6 (recall) and 72.2 (F1). (The other systems did not report B^3.) Interestingly, the rule-based MLN (**RULE**) sufficed to outperform Haghighi & Klein (2007), and by using better heads and the apposition and predicate-nominal rules (**RULE-HAN**), it outperformed McCallum & Wellner (2005), the supervised system. The MLNs with learning (**MLN-30** and **MLN-HAN**), on the other hand, substantially outperformed the corresponding

Table 4.2: Comparison of coreference results in MUC scores on the ACE-2004 (English) datasets.

EN-BNEWS	Prec.	Rec.	F1
H&K	63.2	61.3	62.3
MLN-HAN	66.8	67.8	67.3
EN-NWIRE	Prec.	Rec.	F1
H&K	66.7	62.3	64.2
MLN-HAN	71.3	70.5	70.9

rule-based ones.

Table 4.2 compares our system to Haghighi & Klein (2007) on the ACE-2004 training set in MUC scores. Again, our system outperformed theirs by a large margin. The B^3 scores of MLN-HAN on the ACE-2004 dataset are 71.6 (precision), 68.4 (recall) and 70.0 (F1) for BNEWS, and 75.7 (precision), 69.2 (recall) and 72.3 (F1) for NWIRE. (Haghighi & Klein (2007) did not report B^3.) Due to license restrictions, we could not compare directly to Culotta *et al.* (2007), who reported overall B^3-F1 of 79.3 on the formal test set.

Table 4.3 compares our system to the recent supervised system by Denis & Baldridge (2007). Our approach tied with Denis & Baldridge (2007) on NWIRE, and was somewhat less accurate on BNEWS and NPAPER. Table 4.4 shows our B^3 scores in this dataset. (Denis & Baldridge (2007) did not report B^3 scores.)

Luo *et al.* (2004) pointed out that one can obtain a very high MUC score simply by lumping all mentions together. B^3 suffers less from this problem but is not perfect. Thus we also report pairwise resolution scores (Table 4.5), the gold number of clusters, and our mean absolute error in the number of clusters (Table 4.6). Systems that simply merge all mentions will have exceedingly low pairwise precision (far below 50%), and very large errors in the number of clusters. Our system has fairly good pairwise precisions and small mean error in the number of clusters, which verifies that our results are sound.

Table 4.3: Comparison of coreference results in MUC scores on the ACE-2 datasets.

BNEWS	Prec.	Rec.	F1
D&B	78.0	62.1	69.2
MLN-HAN	68.3	66.6	67.4
NWIRE	Prec.	Rec.	F1
D&B	75.8	60.8	67.5
MLN-HAN	67.7	67.3	67.4
NPAPER	Prec.	Rec.	F1
D&B	77.6	68.0	72.5
MLN-HAN	69.2	71.7	70.4

Table 4.4: Our coreference results in B^3 scores on the ACE-2 datasets.

Section	Prec.	Rec.	F1
BNEWS	70.3	65.3	67.7
NWIRE	74.7	68.8	71.6
NPAPER	70.0	66.5	68.2

4.5.4 Error Analysis

Many of our system's remaining errors involve nominals. Additional features should be considered to distinguish mentions that have the same head but are different entities. For pronouns, many remaining errors can be corrected using linguistic knowledge like binding theory and salience hierarchy. Our heuristics for identifying appositives and predicate nominals also make many errors, which often can be fixed with additional name entity recognition capabilities (e.g., given "Mike Sullivan, VOA News", it helps to know that the former is a person and the latter an organization). The most challenging case involves phrases with

Table 4.5: Our coreference results in precision, recall, and F1 for pairwise resolution.

Pairwise	Prec.	Rec.	F1
MUC-6	63.0	57.0	59.9
EN-BNEWS	51.2	36.4	42.5
EN-NWIRE	62.6	38.9	48.0
BNEWS	44.6	32.3	37.5
NWIRE	59.7	42.1	49.4
NPAPER	64.3	43.6	52.0

Table 4.6: Average gold number of clusters per document vs. the mean absolute error of our system.

# Clusters	MUC-6	EN-BN	EN-NW
Gold	15.4	22.3	37.2
Mean Error	4.7	3.0	4.8
# Clusters	BNEWS	NWIRE	NPAPER
Gold	20.4	39.2	55.2
Mean Error	2.5	5.6	6.6

different heads that are both proper nouns (e.g., “Mr. Bush” and “the White House”). Handling these cases requires domain knowledge and/or more powerful joint inference.

4.6 Discussion

In our coreference MLN, many formulas use conjunction rather than implication; e.g., the head prediction rule

$$\texttt{InClust}(\texttt{m}, +\texttt{c}) \wedge \texttt{Head}(\texttt{m}, +\texttt{t}).$$

At the first sight, this might seem odd from the logical inference viewpoint. This also raises the practical question: when should a conjunction be used instead of an implication?

We first note that a formula like the head prediction rule is not a logical constraint but rather represents probabilistic dependencies. In particular, it corresponds to multiple formulas of the form of $\texttt{InClust}(\texttt{m}, \texttt{C}_\texttt{i}) \wedge \texttt{Head}(\texttt{m}, \texttt{T}_\texttt{j})$, for all constants $\texttt{C}_\texttt{i}$'s and $\texttt{T}_\texttt{j}$'s, with a separate weight learned for each. With conjunction, these formulas are mutually exclusive and their sets of true groundings do not overlap. If the conjunction is replaced with implication, however, the sets of true groundings overlap substantially among the formulas, and their weights interact in a much more tightly coupled way. Empirically, we found that this makes weight learning unnecessarily harder, leading to poorer results.

To choose between conjunction and implication, we find the following rule of the thumb useful. If a formula captures logical or near-deterministic dependency, then by no mean should we convert the implication into conjunction. However, if a set of formulas model soft correlation, then it is often better to make them mutually exclusive by using conjunction.

Another natural question may rise regarding the formulation of our unsupervised learning objective. In Section 4.4, we separate the observed features into two types: those in X that are always conditioned on, and those in Y whose likelihood is to be maximized. This makes our unsupervised learning partially discriminative. We found that this led to better performance for the same reason that discriminative learning generally improves accuracy in supervised learning. In particular, the features in X ($\texttt{IsPrn}, \texttt{Appo}, \texttt{PredNom}$) are those for which our MLN does not model their occurrences in a way that can help coreference resolution.

4.7 Summary

In this chapter, we applied Markov logic to develop the first unsupervised coreference resolution system that is as accurate as supervised systems. Markov logic facilitates joint inference among mentions and the use of relations like apposition and predicate nominals. It also makes it easy to extend the current system with additional linguistic and world knowledge.

Chapter 5
UNSUPERVISED SEMANTIC PARSING

5.1 Introduction

In the previous two chapters, we consider the tasks of information extraction and coreference resolution, which have rather confined goals compared to general machine reading. In particular, we want to extract arbitrary entities and relations, not just a few prespecified ones as in information extraction. To do this, we need to resolve variations among arbitrary synonymous expressions, not just coreferring noun phrases as in coreference resolution. *Semantic parsing* is a key task for achieving general machine reading. The goal is to map text to canonical meaning representations. This contrasts with semantic role labeling [13] and other forms of shallow semantic processing, which do not aim to produce complete formal meanings.

Traditionally, semantic parsers were constructed manually, but this is too costly and brittle. Recently, a number of machine learning approaches have been proposed [119, 76]. However, they are supervised, and providing the target logical form for each sentence is costly and difficult to do consistently and with high quality. More importantly, they assume that the target meaning representations (e.g., the sets of predicates and objects) are readily available, which is a questionable assumption in general domains.[1] Unsupervised approaches have been applied to shallow semantic tasks (e.g., paraphrasing [67], information extraction [8]), but not to semantic parsing.

In this chapter, we develop the first unsupervised approach to semantic parsing, using Markov logic. Our USP system starts by clustering tokens of the same type, and then recursively clusters expressions whose subexpressions belong to the same clusters. Experiments on a biomedical corpus show that this approach is able to successfully translate syntactic variations into a logical representation of their common meaning (e.g., USP learns to map

[1]For example, imagine trying to identify all objects and predicates for PubMed or Wall Street Journal.

active and passive voice to the same logical form, etc.). This in turn allows it to correctly answer many more questions than systems based on TextRunner [8] and DIRT [67].

5.2 Related Work

5.2.1 Semantic Parsing

The standard language for formal meaning representation is first-order logic. A term is any expression representing an object in the domain. An atomic formula or atom is a predicate symbol applied to a tuple of terms. Formulas are recursively constructed from atomic formulas using logical connectives and quantifiers. A *lexical entry* defines the logical form for a lexical item (e.g., a word). The semantic parse of a sentence is derived by starting with logical forms in the lexical entries and recursively composing the meaning of larger fragments from their parts. In traditional approaches, the lexical entries and meaning-composition rules are both manually constructed. Below are sample rules in a definite clause grammar (DCG) for parsing the sentence: "Microsoft acquired Farecast".

$$
\begin{aligned}
&Verb[\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{ACQUIRE}(\mathtt{x},\mathtt{y})] \rightarrow acquired\\
&NP[\mathtt{MICROSOFT}] \rightarrow Microsoft\\
&NP[\mathtt{FARECAST}] \rightarrow Farecast\\
&VP[\mathtt{rel(obj)}] \rightarrow Verb[\mathtt{rel}] \quad NP[\mathtt{obj}]\\
&S[\mathtt{rel(obj)}] \rightarrow NP[\mathtt{obj}] \quad VP[\mathtt{rel}]
\end{aligned}
$$

The first three lines are lexical entries. They are fired upon seeing the individual words. For example, the first rule applies to the word "acquired" and generates syntactic category *Verb* with the meaning $\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{ACQUIRE}(\mathtt{x},\mathtt{y})$ that represents the acquisition relation. Here, we use the standard lambda-calculus notation, where $\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{ACQUIRE}(\mathtt{x},\mathtt{y})$ represents a function that is true for any $(\mathtt{x},\mathtt{y})$-pair such that $\mathtt{ACQUIRE}(\mathtt{x},\mathtt{y})$ holds. The last two rules compose the meanings of sub-parts into that of the larger part. For example, after the first and third rules are fired, the fourth rule fires and generates $VP[\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{ACQUIRE}(\mathtt{x},\mathtt{y})(\mathtt{FARECAST}]$; this meaning simplifies to $\lambda \mathtt{x}.\mathtt{ACQUIRE}(\mathtt{x},\mathtt{FARECAST})$ by the *λ-reduction rule*, which substitutes the argument for a variable in a functional application.

A major challenge to semantic parsing is syntactic variations of the same meaning, which abound in natural languages. For example, the aforementioned sentence can be rephrased as "Microsoft Corporation bought Farecast," "Farecast was purchased by the Redmond software giant," etc. Manually encoding all these variations into the grammar is tedious and error-prone. Supervised semantic parsing addresses this issue by learning to construct the grammar automatically from sample meaning annotations [76]. Existing approaches differ in the meaning representation languages they use and the amount of annotation required. In the approach of Zettlemoyer and Collins (2005), the training data consists of sentences paired with their meanings in lambda form. A probabilistic combinatory categorial grammar (PCCG) is learned using a log-linear model, where the probability of the final logical form L and meaning-derivation tree T conditioned on the sentence S is

$$P(L, T|S) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(L, T, S)\right)$$

where Z is the normalization constant and f_i are the feature functions with weights w_i. Candidate lexical entries are generated by a domain-specific procedure based on the target logical forms.

The major limitation of supervised approaches is that they require meaning annotations for example sentences. Even in a restricted domain, doing this consistently and with high quality requires nontrivial effort. For unrestricted text, the complexity and subjectivity of annotation render it essentially infeasible; even pre-specifying the target predicates and objects is very difficult. Therefore, to apply semantic parsing beyond limited domains, it is crucial to develop unsupervised methods that do not rely on labeled meanings.

In the past, unsupervised approaches have been applied to some semantic tasks, but not to semantic parsing. For example, DIRT [67] learns paraphrases of binary relations based on distributional similarity of their arguments; TextRunner [8] automatically extracts relational triples in open domains using a self-trained extractor; SNE applies relational clustering to generate a semantic network from TextRunner triples [54]. While these systems illustrate the promise of unsupervised methods, the semantic content they extract is nonetheless shallow and does not constitute the complete formal meaning that can be obtained by a semantic parser.

Another issue is that existing approaches to semantic parsing learn to parse syntax and semantics together.[2] The drawback is that the complexity in syntactic processing is coupled with semantic parsing and makes the latter even harder. For example, when applying their approach to a different domain with somewhat less rigid syntax, Zettlemoyer and Collins (2007) need to introduce new combinators and new forms of candidate lexical entries [120]. Ideally, we should leverage the enormous progress made in syntactic parsing and generate semantic parses directly from syntactic analysis.

5.3 An MLN for Unsupervised Semantic Parsing

Unsupervised semantic parsing (**USP**) rests on three key ideas. First, the target predicate and object constants, which are pre-specified in supervised semantic parsing, can be viewed as clusters of syntactic variations of the same meaning, and can be learned from data. For example, `ACQUIRE` represents the acquisition relation, and can be viewed as the cluster of different forms for expressing this relation, such as "acquired", "bought", "purchased"; `Microsoft` represents the company Microsoft, and can be viewed as the cluster of "Microsoft", "the Redmond software giant", etc.

Second, the identification and clustering of candidate forms are integrated with the learning for meaning composition, where forms that are used in composition with the same forms are encouraged to cluster together, and so are forms that are composed of the same subforms. This amounts to recursive relational clustering, a novel form of relational clustering, where clustering is done not just on fixed elements in relational tuples, but on arbitrary forms that are built up recursively.

Third, while most existing approaches (manual or supervised learning) learn to parse both syntax and semantics, unsupervised semantic parsing starts directly from syntactic analyses and focuses solely on translating them to semantic content. This enables us to leverage advanced syntactic parsers and (indirectly) the available rich resources for them. More importantly, it separates the complexity in syntactic analysis from the semantic one, and makes the latter much easier to perform. In particular, meaning composition does not require domain-specific procedures for generating candidate lexicons, as is often needed by

[2]The only exception that we are aware of is Ge and Mooney (2009) [31].

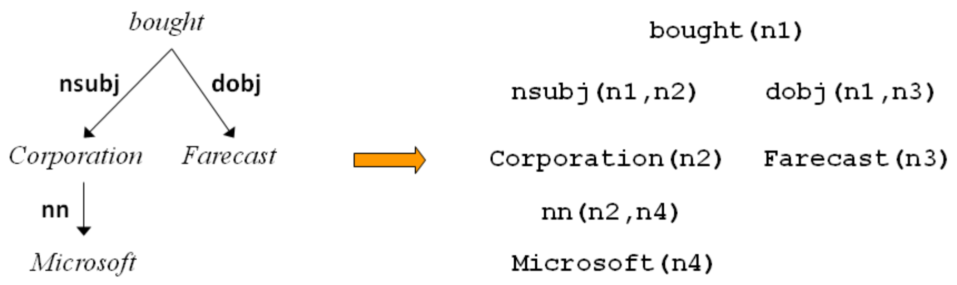

Figure 5.1: Left: the dependency tree for the sentence "Microsoft Corporation bought Farecast". Right: the corresponding quasi-logical form (QLF).

supervised methods.

The input to our USP system consists of dependency trees of training sentences. Compared to phrase-structure syntax, dependency trees are the more appropriate starting point for semantic processing, as they already exhibit much of the relation-argument structure at the lexical level.

USP first uses a deterministic procedure to convert dependency trees into quasi-logical forms (QLFs). The QLFs and their sub-formulas have natural lambda forms, as will be described later. Starting with clusters of lambda forms at the atom level, USP recursively builds up clusters of larger lambda forms. The final output is a probability distribution over lambda-form clusters and their compositions, as well as the MAP semantic parses of training sentences.

In the remainder of the section, we describe the details of USP. We first present the procedure for generating QLFs from dependency trees. We then introduce their lambda forms and clusters, and show how semantic parsing works in this setting. Finally, we present the Markov logic network (MLN) used by USP. In the next section, we present efficient algorithms for learning and inference with this MLN.

5.3.1 Derivation of Quasi-Logical Forms

A *dependency tree* is a tree where nodes are words and edges are dependency labels. Figure 5.1 shows an example for the sentence "Microsoft Corporation bought Farecast" in Stanford

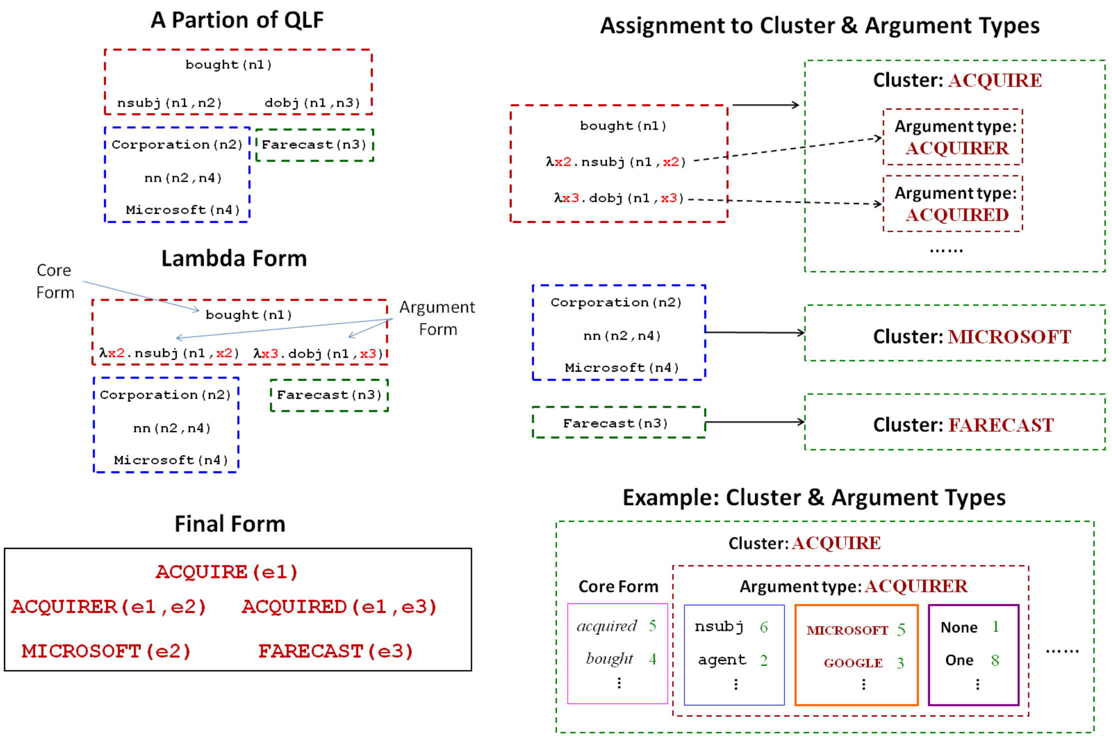

Figure 5.2: An example semantic parse. Top left: a partition that divides the QLF into three sub-formulas and the corresponding lambda forms. Top right: an assignment to clusters and argument types. Bottom left: the final form of the semantic parse. Bottom right: an example cluster with parameters (the numbers are unnormalized multinomial counts).

dependencies [21]. To derive the QLF, we convert each node to an unary atom with the predicate being the lemma plus POS tag (below, we still use the word for simplicity), and each edge to a binary atom with the predicate being the dependency label. For example, in Figure 5.1 the node for Corporation becomes $\texttt{Corporation}(n_2)$ and the subject dependency becomes $\texttt{subj}(\texttt{n1},\texttt{n2})$. Here, the $\texttt{n}_\texttt{i}$'s are Skolem constants indexed by the nodes. The QLF for a sentence is the conjunction of the atoms for the nodes and edges, e.g., the sentence above will become $\texttt{bought}(n_1) \wedge \texttt{Corporation}(n_2) \wedge \texttt{Farecast}(n_3) \wedge \texttt{Microsoft}(n_4) \wedge \texttt{nsubj}(n_1, n_2) \wedge \texttt{dobj}(n_1, n_3) \wedge \texttt{nn}(n_2, n_4)$.

5.3.2 Lambda-Form Clusters and Semantic Parsing in USP

A semantic parse consists of a partition of the QLF and an assignment of the parts to meaning-unit clusters corresponding to relations and objects. We will go through the details using a running example in Figure 5.2. Here, the partition divides the QLF into three subformulas

$$\mathtt{p_1} = \mathtt{bought(n_1)} \wedge \mathtt{nsubj(n_1, n_2)} \wedge \mathtt{dobj(n_1, n_3)}$$

$$\mathtt{p_2} = \mathtt{Corporation(n_2)} \wedge \mathtt{Microsoft(n_4)} \wedge \mathtt{nn(n_2, n_4)}$$

$$\mathtt{p_3} = \mathtt{Farecast(n_3)}$$

USP places no restriction on the partitions, which ensures the maximum flexibility in learning. In particular, lexical entries are no longer limited to be adjacent words as in Zettlemoyer and Collins (2005), but can be arbitrary fragments in a dependency tree.

Given a sub-formula, some atoms signify the core meaning of this unit, while others refer to arguments. For example, in $\mathtt{p_1}$, $\mathtt{bought(n_1)}$ signifies an acquisition event, whereas $\mathtt{nsubj(n_1, n_2)}$ and $\mathtt{dobj(n_1, n_3)}$ lead to the arguments of the acquirer and the acquired, as indexed by $\mathtt{n_2, n_3}$, respectively. These arguments are defined elsewhere (in this case, in the unary atoms $\mathtt{Corporation(n_2), Farecast(n_3)}$), and the subformula $\mathtt{p_1}$ can be viewed as a function that can produce the final form given the arguments as input. We thus follow the standard approach in semantics by representing the sub-formulas in lambda calculus. Specifically, the lambda form of a sub-formula $\mathtt{p}$ is derived by replacing every Skolem constant $\mathtt{n_i}$ that does not appear in any unary atom in $\mathtt{p}$ with a unique lambda variable $\mathtt{x_i}$. The lambda form is then further decomposed into the *core form*, which does not contain any lambda variable (e.g., $\mathtt{bought(n_1)}$), and the *argument forms*, each of which contains a single lambda variable (e.g., $\lambda \mathtt{x_2.nsubj(n_1, x_2)}$ and $\lambda \mathtt{x_3.dobj(n_1, x_3)}$).

A *lambda-form cluster* is a set of semantically interchangeable lambda forms. For example, to express the acquisition event, we can use any form in the cluster `ACQUIRE`, such as "acquired" and "bought". To refer to Microsoft, we can use any form in the cluster `MICROSOFT`, such as "Microsoft Corporation" and "the Redmond software giant". Each lambda-form cluster may contain some number of *argument types*, which cluster distinct forms of the same argument in a relation. For example, `ACQUIRER` is an argument type for `ACQUIRE` that refers to the acquirer argument. There are variations in the syntactic form for an argument type. For example, in Stanford dependencies, the subject of a verb uses the dependency `nsubj` in the active voice, but `agent` in passive, and both can be used in an reference to the acquirer. There are other variations as well. For example, the acquirer may not be mentioned at all, as in "Farecast was acquired for over 100 million". Additionally, the likelihood of being an acquirer differs. For example, a stone doesn't acquire things, while big companies like Microsoft are much more likely to be the acquirer than smaller ones. (Selectional preference captures exactly this kind of variations, except that past works on selectional preference focus on the case where the core meaning is expressed by a word sense rather than arbitrary meaning units.) In sum, a lambda-form clusters abstract away variations in representing the core meaning, whereas an argument type abstracts away variations in referencing a particular kind of argument.

Given a lambda form in cluster $\mathtt{T}$ with arguments of types $\mathtt{A_1}, \cdots, \mathtt{A_k}$, its *abstract lambda form* is given by $\lambda \mathtt{x_1} \cdots \lambda \mathtt{x_k}.\mathtt{T(e)} \wedge \bigwedge_{\mathtt{i}=1}^{\mathtt{k}} \mathtt{A_i(e, x_i)}$. The mapping of a raw form to a cluster or argument type corresponds to a semantic grammar rule. In USP, semantic parsing is conducted by partitioning the QLF to sub-formulas, and then mapping the core forms to clusters and argument forms to argument types by invoking semantic grammar rules. The final form of the semantic parse is obtained by composing the abstract lambda forms of sub-formulas using λ-reduction. Essentially, the USP model defines a probability distribution over the semantic parses by assigning them scores according to the grammar rules and their parameters. The learning problem in USP then amounts to learning the grammar rules and their parameters using the generic templates defined in Markov logic.

USP does not distinguish between relations and objects among meaning units. This enables a unified approach to handle higher-order relations that take other relations as arguments.

5.3.3 The USP MLN

Formally, for a QLF Q, a semantic parse L partitions Q into parts $\mathtt{p_1}, \mathtt{p_2}, \cdots, \mathtt{p_n}$; each part $\mathtt{p}$ is assigned to some lambda-form cluster $\mathtt{c}$, and is further partitioned into core form $\mathtt{f}$ and argument forms $\mathtt{f_1}, \cdots, \mathtt{f_k}$; each argument form is assigned to an argument type $\mathtt{a}$ in

c. The USP MLN defines a joint probability distribution over Q and L by modeling the distributions over forms and arguments given the cluster or argument type.

The USP MLN uses second-order Markov logic where the constants and variables can represent arbitrary lambda forms corresponding to atoms and formulas in the QLF. They should not be confused with the predicates and formulas in the USP MLN.

To model distributions over lambda forms, we introduce the predicates $\texttt{Form}(\texttt{p}, \texttt{f}!)$ and $\texttt{ArgForm}(\texttt{p}, \texttt{i}, \texttt{f}!)$, where p is a part, i is the index of an argument, and f is a QLF subformula. $\texttt{Form}(\texttt{p}, \texttt{f})$ is true iff part p has core form f, and $\texttt{ArgForm}(\texttt{p}, \texttt{i}, \texttt{f})$ is true iff the ith argument in p has form f.[3] The "f!" notation signifies that each part or argument can have only one form.

In our running example in Figure 5.2, there are three parts in the given partition, $\texttt{p}_1, \texttt{p}_2, \texttt{p}_3$. The true Form, ArgForm atoms are determined by the partition, and in this case they are

$$\begin{gathered}
\texttt{Form}(\texttt{p}_1, \text{``}\texttt{bought}(\texttt{n}_1)\text{''}) \\
\texttt{Form}(\texttt{p}_2, \text{``}\texttt{Corporation}(\texttt{n}_2) \land \texttt{Microsoft}(\texttt{n}_4)\text{''}) \\
\texttt{Form}(\texttt{p}_3, \text{``}\texttt{Farecast}(\texttt{n}_3)\text{''}) \\
\texttt{ArgForm}(\texttt{p}_1, 2, \text{``}\lambda \texttt{x}_2.\texttt{nsubj}(\texttt{n}_1, \texttt{x}_2)\text{''}) \\
\texttt{ArgForm}(\texttt{p}_1, 3, \text{``}\lambda \texttt{x}_3.\texttt{dobj}(\texttt{n}_1, \texttt{x}_3)\text{''})
\end{gathered}$$

To model distributions over arguments, we introduce three more predicates: $\texttt{ArgType}(\texttt{p}, \texttt{i}, \texttt{a}!)$ signifies that the ith argument of p is assigned to argument type a; $\texttt{Arg}(\texttt{p}, \texttt{i}, \texttt{p}')$ signifies that the ith argument of p is $\texttt{p}'$; $\texttt{Number}(\texttt{p}, \texttt{a}, \texttt{n})$ signifies that there are n arguments of p that are assigned to type a. The truth value of $\texttt{Number}(\texttt{p}, \texttt{a}, \texttt{n})$ is determined by the ArgType atoms.

In our running example, the ArgType, ArgForm atoms are determined by the semantic parse (specifically, the partition and argument type assignment). In this case, the true

[3]There are hard constraints to guarantee that these assignments form a legal partition. We omit them for simplicity.

atoms are

$$\texttt{Arg}(\texttt{p}_1, 2, \texttt{p}_2)$$

$$\texttt{Arg}(\texttt{p}_1, 3, \texttt{p}_3)$$

$$\texttt{ArgType}(\texttt{p}_1, 2, \texttt{ACQUIRER})$$

$$\texttt{ArgType}(\texttt{p}_1, 3, \texttt{ACQUIRED})$$

$$\texttt{Number}(\texttt{p}_1, \texttt{ACQUIRER}, 1)$$

$$\texttt{Number}(\texttt{p}_1, \texttt{ACQUIRED}, 1)$$

Unsupervised semantic parsing can be captured by four formulas:

$$\texttt{p} \in +\texttt{c} \wedge \texttt{Form}(\texttt{p}, +\texttt{f})$$

$$\texttt{ArgType}(\texttt{p}, \texttt{i}, +\texttt{a}) \wedge \texttt{ArgForm}(\texttt{p}, \texttt{i}, +\texttt{f})$$

$$\texttt{Arg}(\texttt{p}, \texttt{i}, \texttt{p}') \wedge \texttt{ArgType}(\texttt{p}, \texttt{i}, +\texttt{a}) \wedge \texttt{p}' \in +\texttt{c}'$$

$$\texttt{Number}(\texttt{p}, +\texttt{a}, +\texttt{n})$$

All free variables are implicitly universally quantified. The "+" notation signifies that the MLN contains an instance of the formula, with a separate weight, for each value combination of the variables with a plus sign.

Essentially, the USP model defines a recursive mixture model. The first formula models the mixture of core forms given the cluster, and the others model the mixtures of argument forms, argument clusters, and argument numbers, respectively, given the argument type. This amounts to a generative story where the model first picks the core form from a cluster (first formula). Then, for each argument type in the cluster, it determines how many instances to instantiate (fourth formula). For each argument instance, it chooses the argument form (second formula) and the cluster of the argument (third formula). The generative story then continues recursively for each argument cluster that is chosen.

To encourage clustering and avoid overfitting, we impose an exponential prior with weight α on the number of parameters.[4]

The MLN above has one problem: it often clusters expressions that are semantically opposite. For example, it clusters antonyms like "elderly/young", "mature/immature". This issue also occurs in other semantic-processing systems (e.g., DIRT). In general, this

[4]Excluding weights of ∞ or $-\infty$, which signify hard constraints.

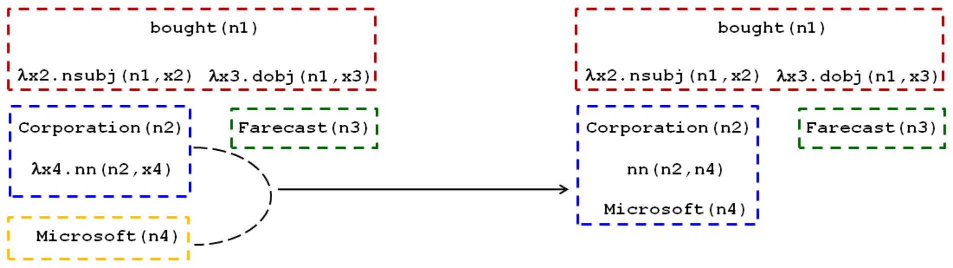

Figure 5.3: The search operator used in USP inference. Left: current partition. Right: new partition after applying lambda-reduction.

is a difficult open problem that only recently has started to receive some attention [75]. Resolving this is not the focus of this paper, but we describe a general heuristic for fixing this problem. We observe that the problem stems from the lack of negative features for discovering meanings in contrast. In natural languages, parallel structures like conjunctions are one such feature.[5] We thus introduce an exponential prior with weight β on the number of conjunctions where the two conjunctive parts are assigned to the same cluster. To detect conjunction, we simply used the Stanford dependencies that begin with "conj". This proves very effective, fixing the majority of the errors in our experiments.

5.4 *Inference and Learning*

Given a sentence and the quasi-logical form Q derived from its dependency tree, the conditional probability for a semantic parse L is given by $Pr(L|Q) \propto \exp\left(\sum_i w_i n_i(L, Q)\right)$. The MAP semantic parse is simply $\arg\max_L \sum_i w_i n_i(L, Q)$. Enumerating all L's is intractable. It is also unnecessary, since most partitions will result in parts whose lambda forms have no cluster they can be assigned to. Instead, USP uses a greedy algorithm to search for the MAP parse. First we introduce some definitions: a partition is called *λ-reducible from* `p` if it can be obtained from the current partition by λ-reducing the part containing `p` with one of its arguments; such a partition is called *feasible* if the core

[5]For example, in the sentence "IL-2 inhibits X in A and induces Y in B", the conjunction between "inhibits" and "induces" suggests that they are different. If "inhibits" and "induces" are indeed synonyms, such a sentence will sound awkward and would probably be rephrased as "IL-2 inhibits X in A and Y in B".

Algorithm 2 USP-Parse(*MLN, QLF*)

```
Form parts for individual atoms in $QLF$ and assign each to its most probable cluster
repeat
  for all parts p in the current partition do
    for all partitions that are $\lambda$-reducible from p and feasible do
      Find the most probable cluster and argument type assignments for the new part
      and its arguments
    end for
  end for
  Change to the new partition and assignments with the highest gain in probability
until none of these improve the probability
return current partition and assignments
```

form of the new part is contained in some cluster. Figure 5.3 illustrates this operator with an example. Consider the QLF of "Microsoft Corporation bought Farecast" and assume that the current partition is $\mathtt{p}_1 = \lambda \mathtt{x}_2\mathtt{x}_3.\mathtt{bought}(\mathtt{n}_1) \wedge \mathtt{nsubj}(\mathtt{n}_1, \mathtt{x}_2) \wedge \mathtt{dobj}(\mathtt{n}_1, \mathtt{x}_3)$, $\mathtt{p}_2 = \lambda \mathtt{x}_4.\mathtt{Corporation}(\mathtt{n}_2) \wedge \mathtt{nn}(\mathtt{n}_2, \mathtt{x}_4)$, $\mathtt{p}_3 = \mathtt{Farecast}(\mathtt{n}_3)$, $\mathtt{p}_4 = \mathtt{Microsoft}(\mathtt{n}_4)$. Then the following partition is λ-reducible by applying λ-reduction to compose $\mathtt{p}_2$ and $\mathtt{p}_4$: $\mathtt{p}_1 = \lambda \mathtt{x}_2\mathtt{x}_3.\mathtt{bought}(\mathtt{n}_1) \wedge \mathtt{nsubj}(\mathtt{n}_1, \mathtt{x}_2) \wedge \mathtt{dobj}(\mathtt{n}_1, \mathtt{x}_3)$, $\mathtt{p}_2' = \mathtt{Corporation}(\mathtt{n}_2) \wedge \mathtt{Microsoft}(\mathtt{n}_4)$, $\mathtt{p}_3 = \mathtt{Farecast}(\mathtt{n}_3)$. Whether this new partition is feasible depends on whether the core form of the new part $\mathtt{Corporation}(\mathtt{n}_2) \wedge \mathtt{Microsoft}(\mathtt{n}_4)$ is contained in some lambda-form cluster.

Algorithm 2 gives pseudo-code for our algorithm. Given part p, finding partitions that are λ-reducible from p and feasible can be done in time $O(ST)$, where S is the size of the clustering in the number of core forms and T is the maximum number of atoms in a core form (by reduction to the unordered subtree matching problem [48]). Inverted indexes (e.g., from p to eligible core forms) are used to further improve the efficiency. For a new part p and a cluster that contains p's core form, there are k^m ways of assigning p's m arguments to the k argument types of the cluster. For larger k and m, this is very expensive. We

Algorithm 3 USP-Learn(*MLN, QLFs*)

```
Create initial clusters and semantic parses
Merge clusters with the same core form
Agenda $\leftarrow \emptyset$
repeat
  for all candidate operations $O$ do
    Score $O$ by log-likelihood improvement
    if score is above a threshold then
      Add $O$ to agenda
    end if
  end for
  Execute the highest scoring operation $O^*$ in the agenda
  Regenerate MAP parses for affected QLFs and update agenda and candidate operations
until agenda is empty
return the MLN with learned weights and the semantic parses
```

therefore approximate it by assigning each argument to the best type, independent of other arguments.

The learning problem in USP is to maximize the log-likelihood of observing the QLFs obtained from the dependency trees, denoted by Q, summing out the unobserved semantic parses:

$$L_\theta(Q) = \log P_\theta(Q) = \log \sum_L P_\theta(Q, L)$$

where L are the semantic parses, θ are the MLN parameters, and $P_\theta(Q, L)$ is the completion likelihood. A serious challenge in unsupervised learning is the identifiability problem (i.e., the optimal parameters are not unique) [64]. This problem is particularly severe for log-linear models with hard constraints, which are common in MLNs. For example, in our USP MLN, conditioned on the fact that $\mathtt{p} \in \mathtt{c}$, there is exactly one value of $\mathtt{f}$ that can satisfy the formula $\mathtt{p} \in \mathtt{c} \wedge \mathtt{Form}(\mathtt{p}, \mathtt{f})$, and if we add some constant number to the weights of

$\mathtt{p} \in \mathtt{c} \wedge \mathtt{Form}(\mathtt{p},\mathtt{f})$ for all $\mathtt{f}$, the probability distribution stays the same.[6] The learner can be easily confused by the infinitely many optima, especially in the early stages. To address this problem, we impose local normalization constraints on specific groups of formulas that are mutually exclusive and exhaustive, i.e., in each group, we require that $\sum_{i=1}^{k} e^{w_i} = 1$, where w_i are the weights of formulas in the group. Grouping is done in such a way as to encourage the intended mixture behaviors. Specifically, for the rule $\mathtt{p} \in +\mathtt{c} \wedge \mathtt{Form}(\mathtt{p},+\mathtt{f})$, all instances given a fixed $\mathtt{c}$ form a group; for each of the remaining three rules, all instances given a fixed $\mathtt{a}$ form a group. Notice that with these constraints the completion likelihood $P(Q, L)$ can be computed in closed form for any L. In particular, each formula group contributes a term equal to the weight of the currently satisfied formula. In addition, the optimal weights that maximize the completion likelihood $P(Q, L)$ can be derived in closed form using empirical relative frequencies. E.g., the optimal weight of $\mathtt{p} \in \mathtt{c} \wedge \mathtt{Form}(\mathtt{p},\mathtt{f})$ is $\log(n_{\mathtt{c},\mathtt{f}}/n_{\mathtt{c}})$, where $n_{\mathtt{c},\mathtt{f}}$ is the number of parts $\mathtt{p}$ that satisfy both $\mathtt{p} \in \mathtt{c}$ and $\mathtt{Form}(\mathtt{p},\mathtt{f})$, and $n_{\mathtt{c}}$ is the number of parts $\mathtt{p}$ that satisfy $\mathtt{p} \in \mathtt{c}$.[7] We leverage this fact for efficient learning in USP.

Another major challenge in USP learning is the summation in the likelihood, which is over all possible semantic parses for a given dependency tree. Even an efficient sampler like MC-SAT [87], as used in Poon & Domingos (2008), would have a hard time generating accurate estimates within a reasonable amount of time. On the other hand, as already noted in the previous section, the lambda-form distribution is generally sparse. Large lambda-forms are rare, as they correspond to complex expressions that are often decomposable into smaller ones. Moreover, while ambiguities are present at the lexical level, they quickly diminish when more words are present. Therefore, a lambda form can usually only belong to a small number of clusters, if not a unique one. We thus simplify the problem by approximating the sum with the mode, and search instead for the L and θ that maximize $\log P_\theta(Q, L)$. Since the optimal weights and log-likelihood can be derived in closed form

[6]Regularizations, e.g., Gaussian priors on weights, alleviate this problem by penalizing large weights, but it remains true that weights within a short range are roughly equivalent.

[7]To see this, notice that for a given $\mathtt{c}$, the total contribution to the completion likelihood from all groundings in its formula group is $\sum_{\mathtt{f}} w_{\mathtt{c},\mathtt{f}} n_{\mathtt{c},\mathtt{f}}$. In addition, $\sum_{\mathtt{f}} n_{\mathtt{c},\mathtt{f}} = n_{\mathtt{c}}$ and there is the local normalization constraint $\sum_{\mathtt{f}} e^{w_{\mathtt{c},\mathtt{f}}} = 1$. The optimal weights $w_{\mathtt{c},\mathtt{f}}$ are easily derived by solving this constrained optimization problem.

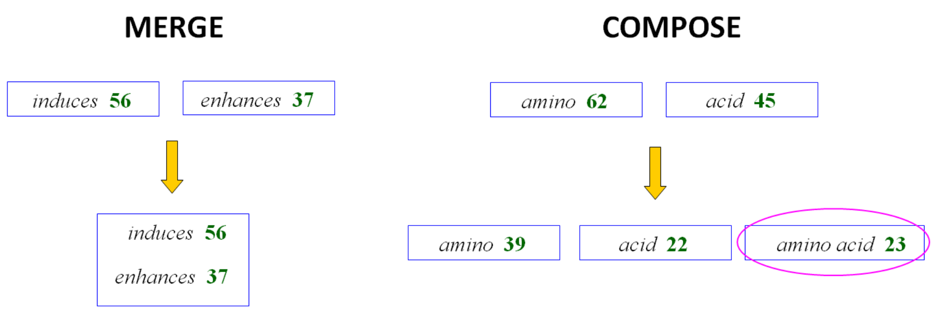

Figure 5.4: Example operations of `MERGE` and `COMPOSE`. For simplicity, only core forms and their unnormalized counts are shown.

given the semantic parses L, we simply search over semantic parses, evaluating them using log-likelihood.

Algorithm 3 gives pseudo-code for our algorithm. The input consists of an MLN without weights and the QLFs for the training sentences. Two operators are used for updating semantic parses. The first is to merge two clusters, denoted by $\texttt{MERGE}(\texttt{C}_1, \texttt{C}_2)$ for clusters $\texttt{C}_1, \texttt{C}_2$, which does the following:

1. Create a new cluster $\texttt{C}$ and add all core forms in $\texttt{C}_1, \texttt{C}_2$ to $\texttt{C}$;
2. Create new argument types for $\texttt{C}$ by merging those in $\texttt{C}_1, \texttt{C}_2$ so as to maximize the log-likelihood;
3. Remove $\texttt{C}_1, \texttt{C}_2$.

Here, merging two argument types refers to pooling their argument forms to create a new argument type. Enumerating all possible ways of creating new argument types is intractable. USP approximates it by considering one type at a time and either creating a new type for it or merging it to types already considered, whichever maximizes the log-likelihood. The types are considered in decreasing order of their numbers of occurrences so that more information is available for each decision. `MERGE` clusters syntactically different expressions whose meanings appear to be the same according to the model.

The second operator is to create a new cluster by composing two existing ones, denoted by $\texttt{COMPOSE}(\texttt{C}_\texttt{R}, \texttt{C}_\texttt{A})$, which does the following:

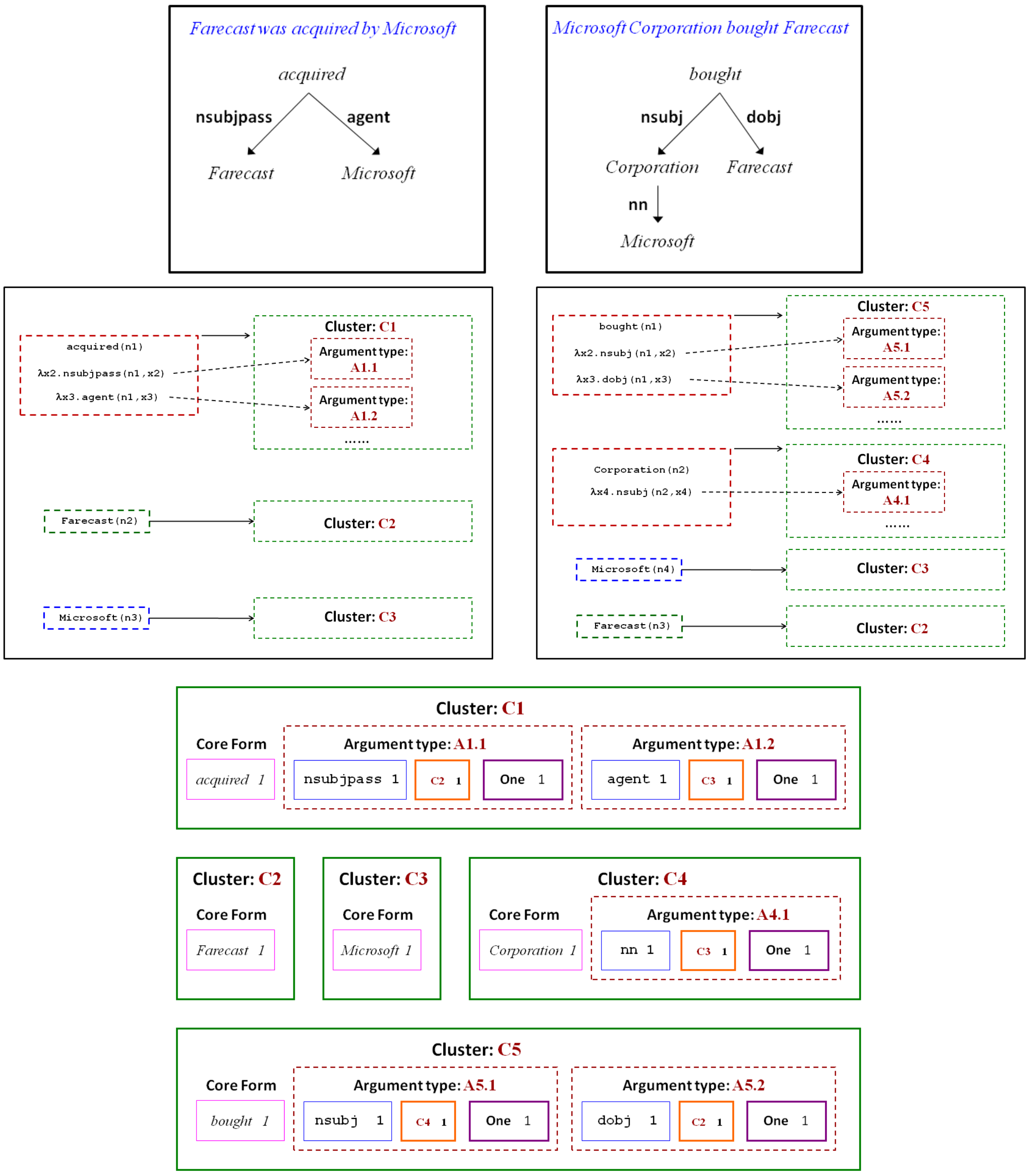

Figure 5.5: Initialization of USP-Learn. Top: a toy corpus with two sentences and their dependency trees. Middle: initial parses by shattering formulas and clustering same forms. Bottom: initial clusters and parameters (in unnormalized multinomial counts).

1. Create a new cluster $\mathtt{C}$;

2. Find all parts $\mathtt{r} \in \mathtt{C_R}, \mathtt{a} \in \mathtt{C_A}$ such that $\mathtt{a}$ is an argument of $\mathtt{r}$, compose them to $\mathtt{r(a)}$ by λ-reduction and add the new part to $\mathtt{C}$;

3. Create new argument types for $\mathtt{C}$ from the argument forms of $\mathtt{r(a)}$ so as to maximize the log-likelihood.

COMPOSE creates clusters of large lambda-forms if they tend to be composed of the same sub-forms (e.g., the lambda form for "the Redmond software giant"). These lambda-forms may later be merged with other clusters (e.g., Microsoft).

See Figure 5.4 for example operations.

At learning time, USP maintains an *agenda* that contains operations that have been evaluated and are pending execution. During initialization, USP forms a part and creates a new cluster for each unary atom $\mathtt{u(n)}$. It also assigns binary atoms of the form $\mathtt{b(n, n')}$ to the part as argument forms and creates a new argument type for each. This forms the initial parses. From these parses, we can derive initial clusters and MAP parameters by merging clusters with the same core form (i.e., the same unary predicate) using MERGE.[8] Figure 5.5 illustrates this process with an example.

In each iteration, all candidate operations are evaluated. To evaluate a candidate operation, USP first updates the MAP parses where the operator is applicable, from which it then updates the clusters and parameters, and calculate the delta in regularized log-likelihood of the MAP parses. These updates are sparse and can be computed very efficiently using inverted indexes.

These candidate operations are then added to the agenda if the improvement in log-likelihood is above a threshold.[9] The operation with the highest score is executed, and the parameters are updated with the new optimal values. The QLFs which contain an affected part are reparsed, and the remaining operations in the agenda whose score might be affected

[8]Word-sense disambiguation can be handled by including a new kind of operator that splits a cluster into subclusters. We leave this to future work.

[9]We currently set it to 10 to favor precision and guard against errors due to inexact estimates.

are re-evaluated. USP terminates when the agenda is empty, and outputs the current MLN parameters and MAP semantic parses.

USP learning uses the same optimization objective as hard EM, and is also guaranteed to find a local optimum since at each step it improves the log-likelihood. It differs from EM in directly optimizing the likelihood instead of a lower bound.

5.5 Experiments

5.5.1 Task

Evaluating unsupervised semantic parsers is difficult, because there is no predefined formal language or gold logical forms for the input sentences. Thus the best way to test them is by using them for the ultimate goal: answering questions based on the input corpus. In this paper, we applied USP to extracting knowledge from biomedical abstracts and evaluated its performance in answering a set of questions that simulate the information needs of biomedical researchers. We used the GENIA dataset [49] as the source for knowledge extraction. It contains 1999 PubMed abstracts and marks all mentions of biomedical entities according to the GENIA ontology, such as cell, protein, and DNA. As a first approximation to the questions a biomedical researcher might ask, we generated a set of two thousand questions on relations between entities. Sample questions are: "What regulates MIP-1alpha?", "What does anti-STAT 1 inhibit?". To simulate the real information need, we sample the relations from the 100 most frequently used verbs (excluding the auxiliary verbs *be*, *have*, and *do*), and sample the entities from those annotated in GENIA, both according to their numbers of occurrences. We evaluated USP by the number of answers it provided and the accuracy as determined by manual labeling.[10]

5.5.2 Systems

Since USP is the first unsupervised semantic parser, conducting a meaningful comparison of it with other systems is not straightforward. Standard question-answering (QA) benchmarks do not provide the most appropriate comparison, because they tend to simul-

[10]The labels and questions are available at http://alchemy.cs.washington.edu/papers/poon09.

taneously emphasize other aspects not directly related to semantic parsing. Moreover, most state-of-the-art QA systems use supervised learning in their key components and/or require domain-specific engineering efforts. The closest available system to USP in aims and capabilities is TextRunner [8], and we compare with it. TextRunner is the state-of-the-art system for open-domain information extraction; its goal is to extract knowledge from text without using supervised labels. Given that a central challenge to semantic parsing is resolving syntactic variations of the same meaning, we also compare with RESOLVER [117], a state-of-the-art unsupervised system based on TextRunner for jointly resolving entities and relations, and DIRT [67], which resolves paraphrases of binary relations. Finally, we also compared to an informed baseline based on keyword matching.

Keyword: We consider a baseline system based on keyword matching. The question substring containing the verb and the available argument is directly matched with the input text, ignoring case and morphology. We consider two ways to derive the answer given a match. The first one (**KW**) simply returns the rest of sentence on the other side of the verb. The second one (**KW-SYN**) is informed by syntax: the answer is extracted from the subject or object of the verb, depending on the question. If the verb does not contain the expected argument, the sentence is ignored.

TextRunner: TextRunner inputs text and outputs relational triples in the form (R, A_1, A_2), where R is the relation string, and A_1, A_2 the argument strings. Given a triple and a question, we first match their relation strings, and then match the strings for the argument that is present in the question. If both match, we return the other argument string in the triple as an answer. We report results when exact match is used (**TR-EXACT**), or when the triple string can contain the question one as a substring (**TR-SUB**).

RESOLVER: RESOLVER [117] inputs TextRunner triples and collectively resolves coreferent relation and argument strings. On the GENIA data, using the default parameters, RESOLVER produces only a few trivial relation clusters and no argument clusters. This is not surprising, since RESOLVER assumes high redundancy in the data, and will discard any strings with fewer than 25 extractions. For a fair comparison, we also ran RESOLVER using all extractions, and manually tuned the parameters based on eyeballing of clustering quality. The best result was obtained with 25 rounds of execution and with the entity multiple set to 200 (the default is 30). To answer questions, the only difference from TextRunner is that a question string can match any string in its cluster. As in TextRunner, we report results for both exact match (**RS-EXACT**) and substring (**RS-SUB**).

DIRT: The DIRT system inputs a path and returns a set of similar paths. To use DIRT in question answering, we queried it to obtain similar paths for the relation of the question, and used these paths while matching sentences. We first used MINIPAR [66] to parse input text using the same dependencies as DIRT. To determine a match, we first check if the sentence contains the question path or one of its DIRT paths. If so, and if the available argument slot in the question is contained in the one in the sentence, it is a match, and we return the other argument slot from the sentence if it is present. Ideally, a fair comparison will require running DIRT on the GENIA text, but we were not able to obtain the source code. We thus resorted to using the latest DIRT database released by the author, which contains paths extracted from a large corpus with more than 1GB of text. In our experiments, we used the top three similar paths, as including more results in very low precision.

USP: We built a system for knowledge extraction and question answering on top of USP. It generated Stanford dependencies [21] from the input text using the Stanford parser, and then fed these to USP-Learn[11], which produced an MLN with learned weights and the MAP semantic parses of the input sentences. These MAP parses formed our knowledge base (KB). To answer questions, the system first parses the questions[12] using USP-Parse with the learned MLN, and then matches the question parse to parses in the KB by testing subsumption (i.e., a question parse matches a KB one iff the former is subsumed by the latter). When a match occurs, our system then looks for arguments of type in accordance with the question. For example, if the question is "What regulates MIP-1alpha?", it searches for the argument type of the relation that contains the argument form "nsubj" for subject. If such an argument exists for the relation part, it will be returned as the answer.

Table 5.1: Comparison of question answering results on the GENIA dataset.

	# Total	# Correct	Accuracy
KW	150	67	45%
KW-SYN	87	67	77%
TR-EXACT	29	23	79%
TR-SUB	152	81	53%
RS-EXACT	53	24	45%
RS-SUB	196	81	41%
DIRT	159	94	59%
USP	**334**	**295**	**88%**

5.5.3 Results

Table 5.1 shows the results for all systems. USP extracted the highest number of answers, almost doubling that of the second highest (RS-SUB). It obtained the highest accuracy at 88%, and the number of correct answers it extracted is three times that of the second highest system. The informed baseline (KW-SYN) did surprisingly well compared to systems other than USP, in terms of accuracy and number of correct answers. TextRunner achieved good accuracy when exact match is used (TR-EXACT), but only obtained a fraction of the answers compared to USP. With substring match, its recall substantially improved, but precision dropped more than 20 points. RESOLVER improved the number of extracted answers by sanctioning more matches based on the clusters it generated. However, most of those additional answers are incorrect due to wrong clustering.[13] DIRT obtained the second highest number of correct answers, but its precision is quite low because the similar paths

[11]α and β are set to -5 and -10.

[12]The question slot is replaced by a dummy word.

[13]We should note that both TextRunner and RESOLVER were developed for extraction from general web text. In future work we plan to evaluate USP in many more domains.

contain many errors.

5.5.4 Qualitative Analysis

Manual inspection shows that USP is able to resolve many nontrivial syntactic variations without user supervision. It consistently resolves the syntactic difference between active and passive voices. It successfully identifies many distinct argument forms that mean the same (e.g., “X stimulates Y” ≈ “Y is stimulated with X”, “expression of X” ≈ “X expression”). It also resolves many nouns correctly and forms meaningful groups of relations. Here are some sample clusters in core forms:

{investigate, examine, evaluate, analyze, study, assay}

{diminish, reduce, decrease, attenuate}

{synthesis, production, secretion, release}

{dramatically, substantially, significantly}

An example question-answer pair, together with the source sentence, is shown below:

Q: What does IL-13 enhance?

A: The 12-lipoxygenase activity of murine macrophages.

Sentence: The data presented here indicate that (1) the 12-lipoxygenase activity of murine macrophages is upregulated in vitro and in vivo by IL-4 and/or IL-13, . . .

5.6 User Guide to the USP Software and Dataset

The USP code and data are available online at `alchemy.cs.washington.edu/usp`. In this section, we will provide step-by-step instructions for using the USP system in unsupervised semantic parsing and question answering. Throughout this section, we assume that you are at the directory where you unpack `usp-code.tar.gz`, and we will use the experiments in this chapter as a running example to show how to apply USP to the GENIA dataset [49] and answer questions about biological entities and relations.

5.6.1 File List

`poon09.pdf`: the paper on unsupervised semantic parsing and the USP system.

`questions.txt:` the questions used in the question-answering evaluation.

`usp.eval:` the answers extracted by the USP system and labels.

`usp-code.tar.gz:` the USP code archive, which includes the following files:

`usp.jar:` Java executable for USP.

`src/:` source code for USP.

`genia/:` default data directory containing the input files for replicating the USP experiments in [90]. It contains raw text and syntactic analyses (morphologies and Stanford dependencies) of 1999 PubMed abstracts in GENIA [49].

`results/:` default directory for the output of USP.

`eval/:` default directory for the question files.

5.6.2 Input To USP

The input to USP consists of syntactic analyses of source text. In particular, USP assumes that the data directory (`genia` in our example) contains three sub-directories:

`text`/ contains the raw text files.

`morph`/ contains input tokens, stems and POS tags.

`dep`/ contains the Stanford dependencies obtained with the format of `collapsedTree`.

5.6.3 Running USP

To run USP with the default directories, run:

```
java -mx20000m -cp usp.jar semantic.Parse genia results
```

This will generate the following files in `results`:

`genia.mln` contains counts tallied from the MAP parse, which implicitly represent the learned parameters. Each cluster starts with a line containing the unique ID along with core forms and counts. It is followed by a number of argument types, each of which consists of three lines: first the ID and counts for various argument numbers, second the argument forms and counts, third the argument clusters and counts.

`genia.parse` contains the MAP semantic parses. Every sentence is partitioned into subformulas, each of which is specified by four lines: first the unique ID (docId+sentenceId+partId) and the text fragment, second the cluster ID and core form, third the ID of the parent subformula and cluster, fourth the argument type ID along with the argument form and ID.

`genia.clustering` contains the core forms of non-unit clusters. Each cluster is represented by a line containing the core forms and counts.

The current implementation requires substantial amount of memory. With a heap size of 20GB, it takes about 20 minutes on the GENIA dataset on a 64-bit linux machine with Intel Xeon 2.3GHz processors. With a smaller heap size (e.g. -mx8000m), the running time is substantially longer (e.g., 80 minutes). Running USP on the Penn-Treebank WSJ data takes about 50 minutes on the same machine and requires about 25GB of memory. Running USP on a 32-bit machine will result in out-of-memory error due to heap size limit. In future work, we will seek to reduce memory requirement by making use of database engine and distributed computing.

The exponential priors can be specified through optional parameters priorNumParam and priorNumConj. For a full list of options, run the program without any parameter.

5.6.4 Question Answering

To replicate the question answering experiment with the learned model and MAP parses, run

```
   java -cp usp.jar eval.USP eval results genia
```

This assumes that `eval/` contains the question files, and `results/` contains the learned model and parses. The output consists of the answers extracted by USP for the questions. Currently, our implementation only handles simple factoid questions as mentioned earlier in the chapter.

5.7 Discussion

Traditional semantic parsers have been confined to well-circumscribed domains. For semantic parsing to be more broadly applicable (e.g., to apply to machine reading from the web), it is imperative to reduce the labeling cost by developing unsupervised or minimally supervised approaches. The USP system is the first attempt in this direction and the experimental results show that it is quite promising. However, much remains to be done.

First, for USP and semantic parsing to have significant impact in web-scale applications such as machine reading, scalability is a key issue. The GENIA dataset used in our experiments contains only thousands of abstracts, which is a far cry from millions in PubMed and billions in the web. While USP could be more accurate than TextRunner, it still lags far behind in scalability compared to TextRunner. Scaling up inference and learning in USP remains a challenging open question.

Second, USP does not start from raw text and requires dependency parses generated by supervised parsers.[14] This limits the applicability of USP to domains for which a good syntactic parser is available. USP is susceptible to errors in syntactic parsing[15] and there is no way to feed semantic information discovered by USP back to syntactic parsing. To solve this problem, we can extend USP to conduct both syntactic and semantic parsing in a joint process. (For example, by combining with unsupervised dependency parsing.) Dur-

[14]However, note that grammar induction or unsupervised syntactic parsing often assumes the input to be POS tags generated by supervised taggers.

[15]However, when we conducted the same experiment using gold dependency parses available in GENIA, there was no increase in accuracy or the number of correct answers.

ing learning, existing resources in syntactic parsing can still be leveraged by treating them as known labels for syntactic categories, similar to the way we incorporated known entity information for pronouns and certain prefixes and suffixes in unsupervised coreference resolution (see Section 4.3.1 and Section 4.4). Similarly, existing knowledge base and ontologies can also be incorporated into USP as known labels for clusters and argument types (and probably with uncertainty to allow for error correction).

Third, USP has not taken advantage of inference to incorporate extracted knowledge into the model. Inference can potentially correct errors in semantic parsing by identifying inconsistent interpretations. It can also infer implicit facts [96], which not only contribute to the set of extracted knowledge, but also provide additional sources of evidence for detecting distributional similarity.

The current USP model also has some known drawbacks in resolving synonymous variations. In particular, USP can not resolve the variations between a verb and its synonymous nominal expression (e.g., “acquire” vs. “acquisition”). This is due to systematic variations in the contexts for these two forms; verbal expressions usually contain more arguments compared to nominal ones, rendering the signal in distributional similarity rather weak. E.g., the patient argument may be realized as an object in a verbal expression, whereas in a nominal expression it is probably realized as a prepositional argument (as in “acquisition of A”), and the agent argument is often omitted in nominal expressions. To overcome this problem, we can extend the model to postulate multiple facets for a meaning unit, each with a distinct set of variations. To compensate for the weak signal from distributional contexts, we can incorporate morphological information to associate a verb with its nominalization.

Finally, there are many linguistic phenomena that USP did not handle as of yet, such as quantifiers, plurals, tense, temporal relations, superlatives, determiners, negation, word senses, etc.[16] Traditional semantic parsers can handle these phenomena if they can be captured in manually engineered rules or learned from labeled examples. USP can leverage these resources as well (if they are available) via semi-supervised learning as in [86]. By

[16]USP handles negation to an extent by recognizing certain function words such as “not”, “n't”. Word sense ambiguity can potentially be handled in USP by starting from tokens rather than types. Namely, the initial clusters of atomic forms only contain individual tokens. Alternatively, we can add a new operator for splitting a cluster that contains different senses of the same expressions.

resolving synonymous variations using unlabeled text, USP can potentially make better use of the available labeled examples since each example effectively becomes multiple examples given the variations.

There is also an interesting scientific question whether some or even all of these regularities can be learned in a mostly unsupervised way. In principle, this appears feasible with the appropriate data and learning bias. For example, if for every statement "A and B" that occurs in the text, the individual statements "A" and "B" can also be found in the text, then the system can potentially discover the meaning of "and" by clustering the first pattern along with the second. Whether this would work in practice is an empirical question. Natural language texts are certainly very different from the aforementioned artificial one. But there are still broad classes of regularities that can be leveraged. For example, facts that can be inferred from common knowledge tend to be omitted. [102] applies this insight to learning generic inference rules, achieving some initial success. Going forward, a particularly exciting direction is to extend the USP model to incorporate discourse structures and pragmatic criteria.

Since the publication of this work, there has been much work that aims to reduce labeling cost in semantic parsing. [109] explores unsupervised semantic parsing in a non-parametric Bayesian framework. [16] and [63] shows that it is possible to learn semantic parsers from question-answer pairs for domains that are well-circumscribed and actionable, such as GeoQuery [118]. [36] further explores learning semantic parsers without the question-answer pairs by incorporating prior knowledge about the domain. In a separate line of work, [5] shows that dialog discourse can be leveraged in semi-supervised learning to compensate for missing labels in semantic parses.

5.8 Summary

This chapter introduces the first unsupervised approach for learning semantic parsers, which represents a significant step forward in learning knowledge representation from data. Our USP system is based on Markov logic, and recursively clusters expressions to abstract away syntactic variations of the same meaning. We constructed an end-to-end machine reading system using USP and successfully applied it to extract knowledge from biomedical text and answer questions.

Chapter 6
UNSUPERVISED ONTOLOGY INDUCTION FROM TEXT

6.1 Introduction

By clustering synonymous meaning units, the USP system can extract general formulas from text and appears to be fairly robust to noise compared to previous state-of-the-art systems such as TextRunner. However, the knowledge extracted by USP is simply a large set of formulas without ontological structure, and the latter is essential for compact representation and efficient reasoning [103]. This also limits the USP extractions to those with substantial evidence in the corpus, and in most corpora most pieces of knowledge are stated only once or a few times, making them very difficult to extract without supervision.

In this chapter, we introduce OntoUSP (Ontological USP), a system that learns an ISA hierarchy over clusters of logical expressions, and populates it by translating sentences to logical form. OntoUSP is encoded in a few formulas of higher-order Markov logic, and can be viewed as extending USP with the capability to perform hierarchical (as opposed to flat) clustering. This clustering is then used to perform hierarchical smoothing (a.k.a. shrinkage), greatly increasing the system's capability to generalize from sparse data.

6.2 Related Work

6.2.1 Ontology Learning

In general, ontology induction (constructing an ontology) and ontology population (mapping textual expressions to concepts and relations in the ontology) remain difficult open problems [103]. Recently, ontology learning has attracted increasing interest in both NLP and semantic-web communities [15, 70], and a number of machine learning approaches have been developed (e.g., Snow et al. (2006) [101], Cimiano (2006) [15], Suchanek et al. (2008,2009) [104, 105], Wu & Weld (2008) [116]). However, they are still limited in several aspects. Most approaches induce and populate a deterministic ontology, which does not capture the inherent uncertainty among the entities and relations. Besides, many of them either bootstrap from heuristic patterns (e.g., Hearst patterns [41]) or build on existing structured or semi-structured knowledge bases (e.g., WordNet [27] and Wikipedia[1]), thus are limited in coverage. Moreover, they often focus on inducing ontology over individual words rather than arbitrarily large meaning units (e.g., idioms, phrasal verbs, etc.). Most importantly, existing approaches typically separate ontology induction from population and knowledge extraction, and pursue each task in a standalone fashion. While computationally efficient, this is suboptimal. The resulted ontology is disconnected from text and requires additional effort to map between the two [110]. In addition, this fails to leverage the intimate connections between the three tasks for joint inference and mutual disambiguation.

Our approach differs from existing ones in two main aspects: we induce a probabilistic ontology from text, and we do so by jointly conducting ontology induction, population, and knowledge extraction. Probabilistic modeling handles uncertainty and noise. A joint approach propagates information among the three tasks, uncovers more implicit information from text, and can potentially work well even in domains not well covered by existing resources like WordNet and Wikipedia. Furthermore, we leverage the ontology for hierarchical smoothing and incorporate this smoothing into the induction process. This facilitates more accurate parameter estimation and better generalization.

Our approach can also leverage existing ontologies and knowledge bases to conduct semi-supervised ontology induction (e.g., by incorporating existing structures as hard constraints or penalizing deviation from them).

6.2.2 Unsupervised Semantic Parsing

Chapter 5 considers the task of semantic parsing and introduced the USP system for unsupervised semantic parsing. See Figure 6.1 for a quick recap by an example. Here, we use a slightly different formulation of the USP and its MLN to facilitate the exposition of OntoUSP. USP inputs dependency trees of sentences and first transforms them into quasi-logical forms (QLFs) by converting each node to a unary atom and each dependency edge

[1]`http : //www.wikipedia.org`

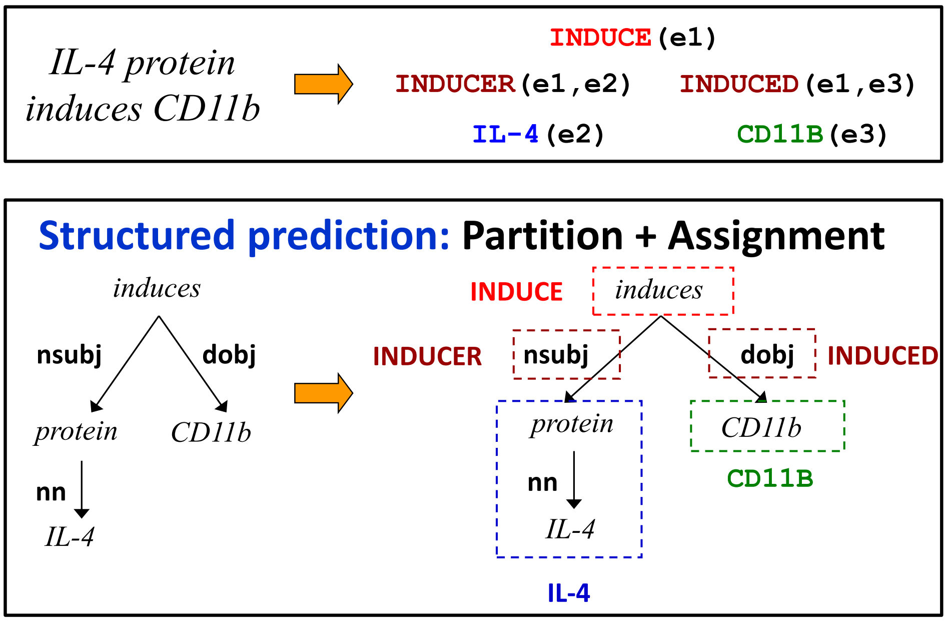

Figure 6.1: An example of semantic parsing. Top: semantic parsing converts an input sentence into logical form in Davidsonian semantics. Bottom: a semantic parse consists of a partition of the dependency tree and an assignment of its parts.

to a binary atom (e.g., the node for "induces" becomes $\texttt{induces}(\texttt{e}_1)$ and the subject dependency becomes $\texttt{nsubj}(\texttt{e}_1, \texttt{e}_2)$, where $\texttt{e}_\texttt{i}$'s are Skolem constants indexed by the nodes.).[2] For each sentence, a semantic parse comprises of a partition of its QLF into subexpressions, each of which has a naturally corresponding lambda form,[3] and an assignment of each subexpression to a lambda-form cluster.

The lambda-form clusters naturally form an ISPART hierarchy. See Figure 6.2 for an example. An *object cluster* corresponds to semantic concepts or relations such as INDUCE, and contains a variable number of *property clusters*. A special property cluster of *core forms* maintains a distribution over variations in lambda forms for expressing this concept or relation. Other property clusters correspond to modifiers or arguments such as INDUCER (the

[2]We call these QLFs because they are not true logical form (the ambiguities are not yet resolved). This is related to but not identical with the definition in Alshawi (1990).

[3]The lambda form is derived by replacing every Skolem constant $\texttt{e}_\texttt{i}$ that does not appear in any unary atom in the subexpression with a lambda variable $\texttt{x}_\texttt{i}$ that is uniquely indexed by the corresponding node i. For example, the lambda form for $\texttt{nsubj}(\texttt{e}_1, \texttt{e}_2)$ is $\lambda \texttt{x}_1 \lambda \texttt{x}_2 . \texttt{nsubj}(\texttt{x}_1, \texttt{x}_2)$.

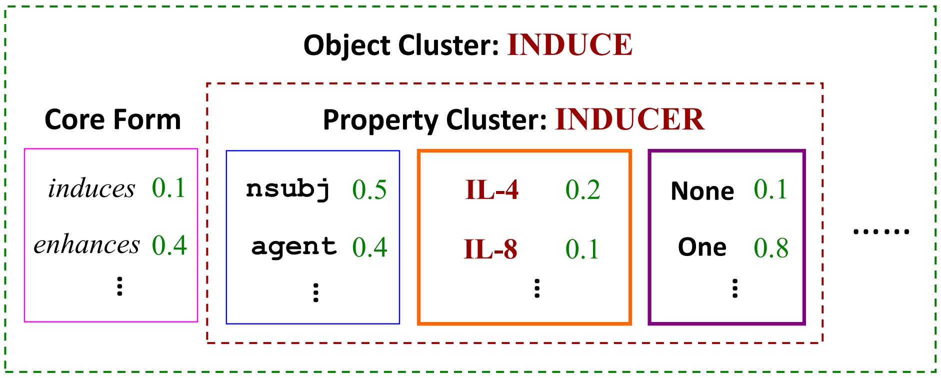

Figure 6.2: An example of object/property clusters: INDUCE contains the core-form property cluster and others, such as the agent argument INDUCER.

agent argument of INDUCE), each of which in turn contains three subclusters of property values: the argument-object subcluster maintains a distribution over object clusters that may occur in this argument (e.g., $\mathtt{IL}-4$), the argument-form subcluster maintains a distribution over lambda forms that corresponds to syntactic variations for this argument (e.g., `nsubj` in active voice and `agent` in passive voice), and the argument-number subcluster maintains a distribution over total numbers of this argument that may occur in a sentence (e.g., zero if the argument is not mentioned).

Effectively, USP simultaneously discovers the lambda-form clusters and an IS-PART hierarchy among them. It does so by recursively combining subexpressions that are composed with or by similar subexpressions. The partition breaks a sentence into subexpressions that are meaning units, and the clustering abstracts away syntactic and lexical variations for the same meaning. This novel form of relational clustering is governed by a joint probability distribution $P(T, L)$ defined in higher-order[4] Markov logic, where T are the input dependency trees, and L the semantic parses. The main predicates are:

$\mathtt{e} \in \mathtt{c}$: expression `e` is assigned to cluster `c`;

$\mathtt{SubExpr}(\mathtt{s}, \mathtt{e})$: `s` is a subexpression of `e`;

[4]Variables can range over arbitrary lambda forms.

$\texttt{HasValue}(\texttt{s},\texttt{v})$**:** $\texttt{s}$ is of value $\texttt{v}$;

$\texttt{IsPart}(\texttt{c},\texttt{i},\texttt{p})$**:** $\texttt{p}$ is the property cluster in object cluster $\texttt{c}$ uniquely indexed by $\texttt{i}$.

In USP, property clusters in different object clusters use distinct index $\texttt{i}$'s. As we will see later, in OntoUSP, property clusters with ISA relation share the same index $\texttt{i}$, which corresponds to a generic semantic frame such as agent and patient.

The probability model of USP can be captured by two formulas:

$$\texttt{x} \in +\texttt{p} \wedge \texttt{HasValue}(\texttt{x}, +\texttt{v})$$
$$\texttt{e} \in \texttt{c} \wedge \texttt{SubExpr}(\texttt{x}, \texttt{e}) \wedge \texttt{x} \in \texttt{p} \Rightarrow \exists^1 \texttt{i}.\texttt{IsPart}(\texttt{c}, \texttt{i}, \texttt{p}).$$

All free variables are implicitly universally quantified. The "+" notation signifies that the MLN contains an instance of the formula, with a separate weight, for each value combination of the variables with a plus sign. The first formula is the core of the model and represents the mixture of property values given the cluster. The second formula ensures that a property cluster must be a part in the corresponding object cluster; it is a hard constraint, as signified by the period at the end.

To encourage clustering, USP imposes an exponential prior over the number of parameters.

To parse a new sentence, USP starts by partitioning the QLF into atomic forms, and then hill-climbs on the probability using a search operator based on lambda reduction until it finds the maximum a posteriori (MAP) parse. During learning, USP starts with clusters of atomic forms, maintains the optimal semantic parses according to current parameters, and hill-climbs on the log-likelihood of observed QLFs using two search operators:

$\texttt{MERGE}(\texttt{c}_1, \texttt{c}_2)$ merges clusters $\texttt{c}_1, \texttt{c}_2$ into a larger cluster $\texttt{c}$ by merging the core-form clusters and argument clusters of $\texttt{c}_1, \texttt{c}_2$, respectively. E.g., $\texttt{c}_1 = \{$"*induce*"$\}$, $\texttt{c}_2 = \{$"*enhance*"$\}$, and $\texttt{c} = \{$"*induce*", "*enhance*"$\}$.

$\texttt{COMPOSE}(\texttt{c}_1, \texttt{c}_2)$ creates a new lambda-form cluster c formed by composing the lambda forms in $\texttt{c}_1, \texttt{c}_2$ into larger ones. E.g., $\texttt{c}_1 = \{$"*amino*"$\}$, $\texttt{c}_2 = \{$"*acid*"$\}$, and $\texttt{c} = \{$"*amino acid*"$\}$.

Each time, USP executes the highest-scored operator and reparses affected sentences using the new parameters. The output contains the optimal lambda-form clusters and parameters, as well as the MAP semantic parses of input sentences.

6.3 An MLN for Unsupervised Ontology Induction

A major limitation of USP is that it either merges two object clusters into one, or leaves them separate. This is suboptimal, because different object clusters may still possess substantial commonalities. Modeling these can help extract more general knowledge and answer many more questions. The best way to capture such commonalities is by forming an ISA hierarchy among the clusters. For example, `INDUCE` and `INHIBIT` are both subconcepts of `REGULATE`. Learning these ISA relations helps answer questions like "What regulates CD11b?", when the text states that "IL-4 induces CD11b" or "AP-1 suppresses CD11b".

For parameter learning, this is also undesirable. Without the hierarchical structure, each cluster estimates its parameters solely based on its own observations, which can be extremely sparse. The better solution is to leverage the hierarchical structure for smoothing (a.k.a. shrinkage [72, 32]). For example, if we learn that "super-induce" is a verb and that in general verbs have active and passive voices, then even though "super-induce" only shows up once in the corpus as in "AP-1 is super-induced by IL-4", by smoothing we can still infer that this probably means the same as "IL-4 super-induces AP-1", which in turn helps answer questions like "What super-induces AP-1".

OntoUSP overcomes the limitations of USP by replacing the flat clustering process with a hierarchical clustering one, and learns an ISA hierarchy of lambda-form clusters in addition to the ISPART one. The output of OntoUSP consists of an ontology, a semantic parser, and the MAP parses. In effect, OntoUSP conducts ontology induction, population, and knowledge extraction in a single integrated process. Specifically, given clusters $\mathtt{c}_1, \mathtt{c}_2$, in addition to merge vs. separate, OntoUSP evaluates a third option called *abstraction*, in which a new object cluster $\mathtt{c}$ is created, and ISA links are added from $\mathtt{c}_\mathtt{i}$ to $\mathtt{c}$; the argument clusters in $\mathtt{c}$ are formed by merging that of $\mathtt{c}_\mathtt{i}$'s.

The OntoUSP MLN can be obtained by modifying the USP MLN with three simple changes. First, we introduce a new predicate $\mathtt{IsA}(\mathtt{c}_1, \mathtt{c}_2)$, which is true if cluster $\mathtt{c}_1$ is a subconcept of $\mathtt{c}_2$. For convenience, we stipulate that $\mathtt{IsA}$ is reflexive (i.e., $\mathtt{IsA}(\mathtt{c},\mathtt{c})$ is true for any $\mathtt{c}$). Second, we add two formulas to the MLN:

$$\mathtt{IsA}(\mathtt{c}_1,\mathtt{c}_2) \wedge \mathtt{IsA}(\mathtt{c}_2,\mathtt{c}_3) \Rightarrow \mathtt{IsA}(\mathtt{c}_1,\mathtt{c}_3).$$

$$\mathtt{IsPart}(\mathtt{c}_1,\mathtt{i}_1,\mathtt{p}_1) \wedge \mathtt{IsPart}(\mathtt{c}_2,\mathtt{i}_2,\mathtt{p}_2) \wedge \mathtt{IsA}(\mathtt{c}_1,\mathtt{c}_2) \Rightarrow (\mathtt{i}_1 = \mathtt{i}_2 \Leftrightarrow \mathtt{IsA}(\mathtt{p}_1,\mathtt{p}_2)).$$

The first formula simply enforces the transitivity of ISA relation. The second formula states that if the ISA relation holds for a pair of object clusters, it also holds between their corresponding property clusters. Both are hard constraints. Third, we introduce hierarchical smoothing into the model by replacing the USP mixture formula

$$\mathtt{x} \in +\mathtt{p} \wedge \mathtt{HasValue}(\mathtt{x},+\mathtt{v})$$

with a new formula

$$\mathtt{ISA}(\mathtt{p}_1,+\mathtt{p}_2)\ \wedge\ \mathtt{x} \in \mathtt{p}_1\ \wedge \mathtt{HasValue}(\mathtt{x},+\mathtt{v})$$

Intuitively, for each $\mathtt{p}_2$, the weight corresponds to the delta in log-probability of $\mathtt{v}$ comparing to the prediction according to all ancestors of $\mathtt{p}_2$. The effect of this change is that now the value $\mathtt{v}$ of a subexpression $\mathtt{x}$ is not solely determined by its property cluster $\mathtt{p}_1$, but is also smoothed by statistics of all $\mathtt{p}_2$ that are super clusters of $\mathtt{p}_1$.

Shrinkage takes place via interaction among the weights of the ISA mixture formula. In particular, if the weights for some property cluster $\mathtt{p}$ are all zero, it means that values in $\mathtt{p}$ are completely predicted by $\mathtt{p}$'s ancestors. In effect, $\mathtt{p}$ is backed off to its parent.

6.4 Inference and Learning

Given the dependency tree T of a sentence, the conditional probability of a semantic parse L is given by $Pr(L|T) \propto \exp\left(\sum_i w_i n_i(T,L)\right)$. The MAP semantic parse is simply $\arg\max_L \sum_i w_i n_i(T,L)$. Directly enumerating all L's is intractable. OntoUSP uses the same inference algorithm as USP by hill-climbing on the probability of L; in each step, OntoUSP evaluates the alternative semantic parses that can be formed by lambda-reducing a current subexpression with one of its arguments. The only difference is that OntoUSP uses a different MLN and so the probabilities and resulting semantic parses may be different. Algorithm 4 gives pseudo-code for OntoUSP's inference algorithm.

Algorithm 4 OntoUSP-Parse(*MLN, T*)

```
Initialize semantic parse $L$ with individual atoms in the $QLF$ of $T$
repeat
  for all subexpressions e in $L$ do
    Evaluate all semantic parses that are lambda-reducible from e
  end for
  $L \leftarrow$ the new semantic parse with the highest gain in probability
until none of these improve the probability
return $L$
```

OntoUSP uses the same learning objective as USP, i.e., to find parameters θ that maximizes the log-likelihood of observing the dependency trees T, summing out the unobserved semantic parses L:

$$L_\theta(T) = \log P_\theta(L) = \log \sum_L P_\theta(T, L)$$

However, the learning problem in OntoUSP is distinct in two important aspects. First, OntoUSP learns in addition an ISA hierarchy among the lambda-form clusters. Second and more importantly, OntoUSP leverages this hierarchy during learning to smooth the parameter estimation of individual clusters, as embodied by the new ISA mixture formula in the OntoUSP MLN.

OntoUSP faces several new challenges unseen in previous hierarchical-smoothing approaches. The ISA hierarchy in OntoUSP is not known in advance, but needs to be learned as well. Similarly, OntoUSP has no known examples of populated facts and rules in the ontology, but has to infer that in the same joint learning process. Finally, OntoUSP does not start from well-formed structured input like relational tuples, but rather directly from raw text. In sum, OntoUSP tackles a very hard problem with exceedingly little aid from user supervision.

To combat these challenges, OntoUSP adopts a novel form of hierarchical smoothing by integrating it with the search process for identifying the hierarchy. Algorithm 5 gives pseudo-code for OntoUSP's learning algorithm. Like USP, OntoUSP approximates the sum

Algorithm 5 OntoUSP-Learn(*MLN, T's*)

```
Initialize with a flat ontology, along with clusters and semantic parses
Merge clusters with the same core form
Agenda ← $\emptyset$
repeat
  for all candidate operations $O$ do
    Score $O$ by log-likelihood improvement
    if score is above a threshold then
      Add $O$ to agenda
    end if
  end for
  Execute the highest scoring operation $O^*$ in the agenda
  Regenerate MAP parses for affected trees and update agenda and candidate operations
until agenda is empty
return the learned ontology and MLN, and the semantic parses
```

over all semantic parses with the most probable parse, and searches for both θ and the MAP semantic parses L that maximize $P_\theta(T, L)$. In addition to `MERGE` and `COMPOSE`, OntoUSP uses a new operator `ABSTRACT`($\mathtt{c}_1, \mathtt{c}_2$), which does the following:

1. Create an *abstract* cluster `c`;
2. Create ISA links from $\mathtt{c}_1, \mathtt{c}_2$ to `c`;
3. Align property clusters of $\mathtt{c}_1$ and $\mathtt{c}_2$; for each aligned pair $\mathtt{p}_1$ and $\mathtt{p}_2$, either merge them into a single property cluster, or create an *abstract* property cluster `p` in `c` and create ISA links from $\mathtt{p}_\mathtt{i}$ to `p`, so as to maximize log-likelihood.

Intuitively, `c` corresponds to a more abstract concept that summarizes similar properties in $\mathtt{c}_\mathtt{i}$'s. See Figure 6.3 for an illustration.

To add a child cluster $\mathtt{c}_2$ to an existing abstract cluster $\mathtt{c}_1$, OntoUSP also uses an operator `ADDCHILD`($\mathtt{c}_1, \mathtt{c}_2$) that does the following:

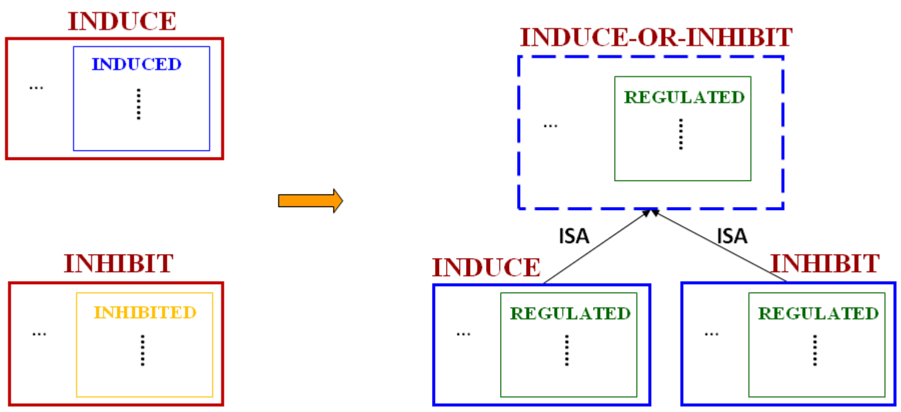

Figure 6.3: The `ABSTRACT` operator illustrated by an example. An abstract cluster is created as the ISA parent and captures substantial similarity such as common argument types.

1. Create an ISA link from $\texttt{c}_2$ to $\texttt{c}_1$;

2. For each property cluster of $\texttt{c}_2$, maximize the log-likelihood by doing one of the following: merge it with a property cluster in an existing child of $\texttt{c}_1$; create ISA link from it to an abstract property cluster in $\texttt{c}$; leave it unchanged.

For efficiency, in both operators, the best option is chosen greedily for each property cluster in $\texttt{c}_2$, in descending order of cluster size.

Notice that once an abstract cluster is created, it could be merged with an existing cluster using `MERGE`. Thus with the new operators, OntoUSP is capable of inducing any ISA hierarchy among abstract and existing clusters. (Of course, the ISA hierarchy it actually induces depends on the data.)

Learning the shrinkage weights has been approached in a variety of ways; examples include EM and cross-validation [72], hierarchical Bayesian methods [32], and maximum entropy with L_1 priors [24]. The past methods either only learn parameters with one or two levels (e.g., in hierarchical Bayes), or requires significant amount of computation (e.g., in EM and in L_1-regularized maxent), while also typically assuming a given hierarchy. In contrast, OntoUSP has to both induce the hierarchy and populate it, with potentially many levels in the induced hierarchy, starting from raw text with little user supervision.

Therefore, OntoUSP simplifies the weight learning problem by adopting standard m-estimation for smoothing. Namely, the weights for cluster $\mathtt{c}$ are set by counting its observations plus m fractional samples from its parent distribution. When $\mathtt{c}$ has few observations, its unreliable statistics can be significantly augmented via the smoothing by its parent (and in turn to a gradually smaller degree by its ancestors). m is a hyperparameter that can be used to trade off bias towards statistics for parent vs oneself.

OntoUSP also needs to balance between two conflicting aspects during learning. On one hand, it should encourage creating abstract clusters to summarize intrinsic commonalities among the children. On the other hand, this needs to be heavily regularized to avoid mistaking noise for the signal. OntoUSP does this by a combination of priors and thresholding. To encourage the induction of higher-level nodes and inheritance, OntoUSP imposes an exponential prior β on the number of *parameter slots*. Each slot corresponds to a distinct property value. A child cluster inherits its parent's slots (and thus avoids the penalty on them). OntoUSP also stipulates that, in an ABSTRACT operation, a new property cluster can be created either as a *concrete* cluster with full parameterization, or as an *abstract* cluster that merely serves for smoothing purposes. To discourage overproposing clusters and ISA links, OntoUSP imposes a large exponential prior γ on the number of concrete clusters created by ABSTRACT. For *abstract* cluster, it sets a cut-off t_p and only allows storing a probability value no less than t_p. Like USP, it also rejects MERGE and COMPOSE operations that improve log-likelihood by less than t_o. These priors and cut-off values can be tuned to control the granularity of the induced ontology and clusters.

Concretely, given semantic parses L, OntoUSP computes the optimal parameters and evaluates the regularized log-likelihood as follows. Let $w_{p_2,v}$ denote the weight of the ISA mixture formula $\mathtt{ISA(p_1, +p_2)} \wedge \mathtt{x} \in \mathtt{p_1} \wedge \mathtt{HasValue(x, +v)}$. For convenience, for each pair of property cluster $\mathtt{c}$ and value $\mathtt{v}$, OntoUSP instead computes and stores $w'_{\mathtt{c,v}} = \sum_{\mathtt{ISA(c,\,a)}} w_{\mathtt{a,v}}$, which sums over all weights for $\mathtt{c}$ and its ancestors. (Thus $w_{\mathtt{c,v}} = w'_{\mathtt{c,v}} - w'_{\mathtt{p,v}}$, where $\mathtt{p}$ is the parent of $\mathtt{c}$.) Like USP, OntoUSP imposes local normalization constraints that enable closed-form estimation of the optimal parameters and likelihood. Specifically, using m-estimation, the optimal $w'_{\mathtt{c,v}}$ is $\log((m \cdot e^{w'_{\mathtt{p,v}}} + n_{\mathtt{c,v}})/(m + n_{\mathtt{c}}))$, where $\mathtt{p}$ is the parent of $\mathtt{c}$ and n is the count. The log-likelihood is $\sum_{\mathtt{c,v}} w'_{\mathtt{c,v}} \cdot n_{\mathtt{c,v}}$, which is then augmented by the priors.

6.5 Experiments

6.5.1 Methodology

Evaluating unsupervised ontology induction is difficult, because there is no gold ontology for comparison. Moreover, our ultimate goal is to aid knowledge acquisition, rather than just inducing an ontology for its own sake. Therefore, we used the same methodology and dataset as the USP paper to evaluate OntoUSP on its capability in knowledge acquisition. Specifically, we applied OntoUSP to extract knowledge from the GENIA dataset [49] and answer questions, and we evaluated it on the number of extracted answers and accuracy. GENIA contains 1999 PubMed abstracts.[5] The question set contains 2000 questions which were created by sampling verbs and entities according to their frequencies in GENIA. Sample questions include “What regulates MIP-1alpha?”, “What does anti-STAT 1 inhibit?”. These simple question types were used to focus the evaluation on the knowledge extraction aspect, rather than engineering for handling special question types and/or reasoning.

6.5.2 Systems

OntoUSP is the first unsupervised approach that synergistically conducts ontology induction, population, and knowledge extraction. The system closest in aim and capability is USP. We thus compared OntoUSP with USP and all other systems evaluated in Chapter 5. Below is a brief description of the systems. (For more details, see Chapter 5.)

Keyword is a baseline system based on keyword matching. It directly matches the question substring containing the verb and the available argument with the input text, ignoring case and morphology. Given a match, two ways to derive the answer were considered: KW simply returns the rest of sentence on the other side of the verb, whereas KW-SYN is informed by syntax and extracts the answer from the subject or object of the verb, depending on the question (if the expected argument is absent, the sentence is ignored).

TextRunner [8] is the state-of-the-art system for open-domain information extraction. It

[5] `http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/home/wiki.cgi`.

inputs text and outputs relational triples in the form (R, A_1, A_2), where R is the relation string, and A_1, A_2 the argument strings. To answer questions, each triple-question pair is considered in turn by first matching their relation strings, and then the available argument strings. If both match, the remaining argument string in the triple is returned as an answer. Results were reported when exact match is used (TR-EXACT), or when the triple strings may contain the question ones as substrings (TR-SUB).

RESOLVER [117] inputs TextRunner triples and collectively resolves coreferent relation and argument strings. To answer questions, the only difference from TextRunner is that a question string can match any string in its cluster. As in TextRunner, results were reported for both exact match (RS-EXACT) and substring (RS-SUB).

DIRT [67] resolves binary relations by inputting a dependency path that signifies the relation and returns a set of similar paths. To use DIRT in question answering, it was queried to obtain similar paths for the relation of the question, which were then used to match sentences.

USP parses the input text using the Stanford dependency parser [51, 21], learns an MLN for semantic parsing from the dependency trees, and outputs this MLN and the MAP semantic parses of the input sentences. These MAP parses formed the knowledge base (KB). To answer questions, USP first parses the questions (with the question slot replaced by a dummy word), and then matches the question parse to parses in the KB by testing subsumption.

OntoUSP uses a similar procedure as USP for extracting knowledge and answering questions, except for two changes. First, USP's learning and parsing algorithms are replaced with OntoUSP-Learn and OntoUSP-Parse, respectively. Second, when OntoUSP matches a question to its KB, it not only considers the lambda-form cluster of the question relation, but also all its sub-clusters. For example, if the question asks: What does IL-2 control? In USP, the answers are derived by searching the knowledge base and matching the lambda-form clusters for "control" and "IL-2". So if "regulate" is in the same cluster as "control", and the KB contains a fact that IL-2 regulates 12-lipoxygenase activity, then "12-lipoxygenase activity" will be returned as the answer. OntoUSP, however, will additionally check all the sub-clusters for the clusters of "control" and "IL-2". So if the cluster of "control" has a sub-cluster that contains "inhibit", and if the KB contains a fact that

Table 6.1: Comparison of question answering results on the GENIA dataset.

	# Total	# Correct	Accuracy
KW	150	67	45%
KW-SYN	87	67	77%
TR-EXACT	29	23	79%
TR-SUB	152	81	53%
RS-EXACT	53	24	45%
RS-SUB	196	81	41%
DIRT	159	94	59%
USP	334	295	88%
OntoUSP	**480**	**435**	**91%**

IL-2 inhibits IKappaBAlpha, then "IKappaBAlpha" will be returned as an answer as well.

6.5.3 Results

Table 6.1 shows the results comparing OntoUSP with other systems. While USP already greatly outperformed other systems in both precision and recall, OntoUSP further substantially improved on the recall of USP, without any loss in precision. In particular, OntoUSP extracted 140 more correct answers than USP, for a gain of 47% in absolute recall. Compared to TextRunner (TR-SUB), OntoUSP gained on precision by 38 points and extracted more than five times of correct answers.

Manual inspection shows that the induced ISA hierarchy is the key for the recall gain. Like USP, OntoUSP discovered the following clusters (in core forms) that represent some of the core concepts in biomedical research:

{regulate, control, govern, modulate}

{induce, enhance, trigger, augment, up-regulate}

{inhibit, block, suppress, prevent, abolish, abrogate, down-regulate}

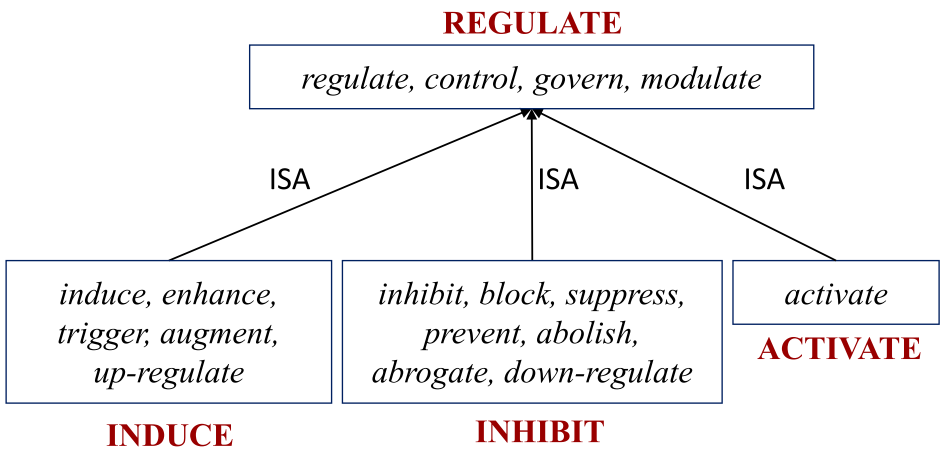
REGULATE
regulate, control, govern, modulate
ISA
ISA
ISA
induce, enhance, trigger, augment, up-regulate
inhibit, block, suppress, prevent, abolish, abrogate, down-regulate
activate
ACTIVATE
INDUCE
INHIBIT

Figure 6.4: A fragment of the induced ISA hierarchy, showing the core forms for each cluster (the cluster labels are added by the authors for illustration purpose).

However, USP formed these as separate clusters, whereas OntoUSP in addition induces ISA relations from the `INDUCE` and `INHIBIT` clusters to the `REGULATE` cluster (Figure 6.4). This allows OntoUSP to answer many more questions that are asked about general regulation events, even though the text states them with specific regulation directions like "induce" or "inhibit". Below is an example question-answer pair output by OntoUSP; neither USP nor any other system were able to extract the necessary knowledge.

Q: What does IL-2 control?

A: The DEX-mediated IkappaBalpha induction.

Sentence: Interestingly, the DEX-mediated IkappaBalpha induction was completely inhibited by IL-2, but not IL-4, in Th1 cells, while the reverse profile was seen in Th2 cells.

OntoUSP also discovered other interesting commonalities among the clusters. For example, both USP and OntoUSP formed a singleton cluster with core form "activate". Although this cluster may appear similar to the `INDUCE` cluster, the data in GENIA does not support merging the two. However, OntoUSP discovered that the `ACTIVATE` cluster, while not completely resolvent with `INDUCE`, shared very similar distributions in their agent arguments. In fact, they are so similar that OntoUSP merges them into a single property cluster. It found that the patient arguments of `INDUCE` and `INHIBIT` are very similar and merged them. In turn, OntoUSP formed ISA links from these three object clusters to `REGULATE`, as well as among their property clusters. Intuitively, this makes sense. The positive- and negative-regulation events, as signified by `INDUCE` and `INHIBIT`, often target similar object entities or processes. However, their agents tend to differ since in one case they are inducers, and in the other they are inhibitors. On the other hand, `ACTIVATE` and `INDUCE` share similar agents since they both signify positive regulation. However, "activate" tends to be used more often when the patient argument is a concrete entity (e.g., cells, genes, proteins), whereas "induce" and others are also used with processes and events (e.g., expressions, inhibition, pathways).

USP was able to resolve common syntactic differences such as active vs. passive voice. However, it does so on the basis of individual verbs, and there is no generalization beyond their clusters. OntoUSP, on the other hand, formed a high-level cluster with two abstract property clusters, corresponding to general agent argument and patient argument. The active-passive alternation is captured in these clusters, and is inherited by all descendant clusters, including many rare verbs like "super-induce" which only occur once in GENIA and for which there is no way that USP could have learned about their active-passive alternations. This illustrates the importance of discovering ISA relations and performing hierarchical smoothing.

6.5.4 Discussion

OntoUSP is a first step towards joint ontology induction and knowledge extraction. The experimental results demonstrate the promise in this direction. However, we also notice some limitations in the current system. While OntoUSP induced meaningful ISA relations among relation clusters like `REGULATE`, `INDUCE`, etc., it was less successful in inducing ISA relations among entity clusters such as specific genes and proteins. This is probably due to the fact that our model only considers local features such as the parent and arguments. A relation is often manifested as verbs and has several arguments, whereas an entity typically appears as an argument of others and has few arguments of its own. As a result, in average, there is less information available for entities than relations. Presumably, we can address this limitation by modeling longer-ranged dependencies such as grandparents, siblings, etc. This is straightforward to do in Markov logic.

OntoUSP also uses a rather elaborate scheme for regularization. We hypothesize that this can be much simplified and improved by adopting a principled framework such as [24].

6.6 Summary

This chapter introduced OntoUSP, the first unsupervised end-to-end system for ontology induction and knowledge extraction from text. OntoUSP builds on the USP semantic parser by adding the capability to form hierarchical clusterings of logical expressions, linked by ISA relations, and using them for hierarchical smoothing. OntoUSP greatly outperformed USP and other state-of-the-art systems in a biomedical knowledge acquisition task.

Chapter 7 CONCLUSION

7.1 Contributions

In this dissertation, we propose a unifying approach for machine reading. At the core of this approach is joint inference and the integration of learning with knowledge representation and deep reasoning. By combining logic and probability, Markov logic provides an ideal framework for this approach: it offers an elegant way to leverage joint inference and incorporate knowledge, and opens up new avenues to compensate for the lack of labeled examples.

It is helpful to contrast our approach with the IBM DeepQA Project [30], which produced the Watson system and achieved spectacular successes in competing with human champions in the trivia game Jeopardy. In Jeopardy, factoid questions are given to the participants, who compete to be the first in producing an accurate answer. The questions can be decomposed into multiple "clues", each of which associates the answer with some atomic fact (factoid). This enables Watson to adopt a rather simplistic approach by converting each clue into a statistical query, and training a classifier to combine the confidence scores to rank candidate answers. To produce the scores, Watson combines hundreds of existing NLP modules and vast amount of relevant resources, and uses past Jeopardy games as labeled examples. This approach is quite effective for solving the Jeopardy game, but it does not address the general challenges in machine reading and question answering. Most saliently, Watson has no deep notions of knowledge or understanding. Consequently, it lacks the capability of inducing meaning representations and acquiring complex knowledge for general machine reading. Nor is it able to conduct long chains of reasoning that is necessary for answering general questions.

In contrast, learning representations for and reasoning with general knowledge are at the core of our approach. In the previous chapters, we took some initial steps in this direction by applying Markov logic to increasing challenging tasks for machine reading, ranging from information extraction to coreference resolution to semantic parsing and end-to-end machine reading. We show that Markov logic can compactly specify very large and complex probabilistic models, and attain state-of-the-art accuracy using the generic learning and inference algorithms (Chapter 3; also see [88]). In addition, by facilitating joint inference and the use of prior knowledge, we show that Markov logic can have even greater impact when labeled examples are absent (Chapter 4; also see [89]). In particular, we can automatically induce a canonical meaning representation (Chapter 5; also see [90]) and learn ontological relations from text (Chapter 6; also see [92]). This results in an end-to-end machine reading system (USP) that does not require labeled examples and substantially outperforms state-of-the-art systems such as TextRunner.

7.2 Limitations of This Work

Despite the progress, we are still a long way from realizing the full potential of Markov logic and solving machine reading.

First, although we have made significant progress in developing efficient inference and learning algorithms, much remains to be done. While Markov logic provides a unifying way to leverage joint inference and knowledge, the extent to which this power can be realized crucially depends on the efficiency and accuracy of inference algorithms. Since exact inference is intractable, the standard paradigm is to conduct approximate inference both in learning (where inference is a subroutine) and at performance time (where inference computes the answer to a query). Learning can be led astray by inaccurate inference results [59]. At performance time, there is little guarantee on the quality and running time. Consequently, although joint inference is intuitively appealing, it is often non-trivial to make it work in practice. Moreover, the current scale of joint inference is minuscule compared to our ultimate goal of reading the entire web. While USP can handle thousands of documents, PubMed has millions of them, and the web has billions.

Second, while USP shows initial promise in learning the knowledge representation, many challenges lie ahead. For example, how can we learn the representation to optimize performance in a specific end task? How can we leverage existing ontologies, databases, and other legacy knowledge engineering works? For machine learning, this problem also represents a new frontier with difficult open questions. For example, how can we tune the hyperparameters in the absence of labeled examples? How do we overcome the limitations of the greedy search procedures in USP learning and inference? Given that there is little labeled data, what is the best way to evaluate our learning algorithms and foster progress?

Third, a key aspect of our vision is to close the loop between the extraction process and reasoning with the extracted knowledge. This has not been achieved yet. Obviously, it would require progress in large-scale joint inference and representation learning. Moreover, we need to model additional linguistic phenomena such as quantifiers, temporal relations, discourse structures, belief and sentiment, etc. And we need to incorporate knowledge and reasoning, which are deeply intertwined with linguistic phenomena. (Consider, for example, the knowledge about an event type and the discourse structure for describing such an instance.)

7.3 Future Directions

Five research directions stand out due to their urgency and potential impact.

7.3.1 Learning for Efficient Inference

The complexity of inference is a key limiting factor for probabilistic graphical modeling in general and Markov logic in particular. By incorporating computation into the model and learning many layers of hidden variables for representing and reusing intermediate computation, we can learn a probabilistic model that guarantees efficient inference while minimizing approximation errors. Essentially, this will automatically construct the dynamic programming structure to approximate data dependencies as close as possible, while subject to the constraint on computation cost that is proportional to the size of the structure. The key challenge is in learning such a deep, hidden architecture. USP successfully learns the nested, hidden structures of semantic analyses. Lessons can potentially be drawn from USP and used in learning the structures for efficient inference.

7.3.2 Scale Joint Inference to the Web

A thought experiment helps illustrate the challenge in scaling up to the web. Suppose we have extracted knowledge from half of the web, accumulated a gigantic knowledge base (KB), and now set off to process the next document. The key to efficient joint inference lies in quickly identifying relevant pieces in the existing KB for processing the new document. At the web scale, even linear-time enumeration of the KB is infeasible. A promising direction is to combine probabilistic ontology induction as used in USP with coarse-to-fine inference [29, 84, 47]. At the abstract levels of the ontology, ambiguities are few and joint inference can be done efficiently. When refining to the next level, we then use previous results to prune away irrelevant branches. This can potentially reduce inference time to logarithmic in the KB size, advancing a key step toward scaling up to the web.

7.3.3 Build a Complete NLP System

Language understanding involves many complex subtasks and is tightly coupled with knowledge and reasoning. To solve machine reading, we need to build a complete NLP system that incorporates all these aspects and benefits from massive scale of joint inference. USP provides a natural starting point for this. But there are many challenges to overcome in addition to efficiency issues. The system needs to model major linguistic phenomena. Moreover, it should leverage knowledge and reasoning in the NLP pipeline, which can happen in two ways. First, the system should incorporate knowledge as it is acquired to resolve ambiguities. Second, texts often omit facts that can be inferred from common-sense knowledge, and the system needs to fill them in using reasoning techniques such as abduction.

7.3.4 Harness Social Computing for Knowledge Extraction

We can leverage social computing to speed up the progress toward the grand goal of constructing a large knowledge base from the web [18]. A machine reading system can bootstrap a user community by serving applications like question answering. It then keeps reading the web, collecting error correction and knowledge donation from users, and using statistical relational learning to incorporate all of these to improve service performance. A positive feedback loop thus forms: as service quality improves, the user base grows and becomes more engaged, which in turn provides more information to help the system improve. To do this, we can build on recent advances such as the collaborative approach to knowledge base construction [93] and the never-ending learning paradigm [11], and use USP to build an initial reading system.

7.3.5 Machine Reading for Machine Science

Scientific knowledge has been accumulated at an explosive rate, with PubMed being a salient example from the biomedical domain. It has become infeasible for scientists to keep abreast of such rapid progress by manual effort. Literature-based discovery (LBD) has produced some surprising discoveries by assembling knowledge across subfields to propose new hypotheses [106]. However, existing LBD methods tend to produce too many false hypotheses, because they only use simple information such as keyword co-occurrence. As a result, they still require much manual effort and domain expertise to filter out false leads. With deeper understanding of the text, machine reading can help produce the next generation of LBD methods and venture a key step toward the exciting prospect of machine science for automating sciences [26].

BIBLIOGRAPHY

[1] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of ACM International Conference on Digital Libraries*, 2000.

[2] James Allen. *Natural Language Understanding.* Benjamin Cummings, 2nd edition, 1994.

[3] Hiyan Alshawi. Resolving quasi logical forms. *Computational Linguistics*, 16:133–144, 1990.

[4] B. Amit and B. Baldwin. Algorithms for scoring coreference chains. In *Proceedings of MUC-7*, 1998.

[5] Yoav Artzi and Luke Zettlemoyer. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 2011.

[6] G. Bakir, T. Hofmann, B. B. Schölkopf, A. Smola, B. Taskar, S. Vishwanathan, and (eds.). *Predicting Structured Data.* MIT Press, Cambridge, MA, 2007.

[7] Michel Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of the Forty Sixth Annual Meeting of the Association for Computational Linguistics*, 2008.

[8] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 2670–2676, Hyderabad, India, 2007. AAAI Press.

[9] Misha Bilenko and Ray Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[10] Razvan Bunescu and Ray Mooney. Collective information extraction with relational markov networks. In *Proceedings of the Forty Second Annual Meeting of the Association for Computational Linguistics*, 2004.

[11] 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 Twenty Fifth National Conference on Artificial Intelligence*, 2010.

[12] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, 2010.

[13] Xavier Carreras and Luis Marquez. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*, pages 89–97, Boston, MA, 2004. ACL.

[14] Eugene Charniak. *Statistical Language Learning.* MIT Press, Cambridge, 1993.

[15] Philipp Cimiano. *Ontology learning and population from text.* Springer, 2006.

[16] James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. Driving semantic parsing from world's response. In *Proceedings of the 2010 Conference on Natural Language Learning*, 2010.

[17] Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[18] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Sean Slattery. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 1999.

[19] A. Culotta, M. Wick, R. Hall, and A. McCallum. First-order probabilistic models for coreference resolution. In *Proceeding of HLT-NAACL*, 2007.

[20] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognizing textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop Recognizing Textual Entailment*, 2005.

[21] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454, Genoa, Italy, 2006. ELRA.

[22] Pascal Denis and Jason Baldridge. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of NAACL-2007*, Rochester, NY, 2007.

[23] Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence.* Morgan & Claypool, San Rafael, CA, 2009.

[24] Miroslav Dudik, David Blei, and Robert Schapire. Hierarchical maximum entropy density estimation. In *Proceedings of the Twenty Fourth International Conference on Machine Learning*, 2007.

[25] O. Etzioni, M. Banko, and M. J. Cafarella. Machine reading. In *Proceedings of the 2007 AAAI Spring Symposium on Machine Reading*, Palo Alto, CA, 2007. AAAI Press.

[26] James Evans and Andrey Rzhetsky. Machine science. *Science*, 329, 2010.

[27] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database.* MIT Press, Cambridge, MA, 1998.

[28] Pedro Felzenszwalb and Ramin Zabih. Dynamic programming and graph algorithms in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4), April 2011.

[29] Pedro F. Felzenszwalb and David McAllester. The generalized A* architecture. *Journal of Artificial Intelligence Research*, 29, 2007.

[30] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building watson: An overview of the deepqa project. *AI Magazine*, 2010.

[31] Ruifang Ge and Raymond J. Mooney. Learning a compositional semantic parser using an existing syntactic parser. In *Proceedings of the Forty Seventh Annual Meeting of the Association for Computational Linguistics*, Singapore, 2009. ACL.

[32] Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models.* Cambridge University Press, 2006.

[33] M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, San Mateo, CA, 1987.

[34] Lise Getoor and Ben Taskar, editors. *Introduction to Statistical Relational Learning.* MIT Press, Cambridge, MA, 2007.

[35] W. R. Gilks, S. Richardson D. J. Spiegelhalter, and eds., editors. *Markov Chain Monte Carlo in Practice.* Chapman and Hall, 1996.

[36] Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. Confidence driven unsupervised semantic parsing. In *Proceedings of the Forty Ninth Annual Meeting of the Association for Computational Linguistics*, 2011.

[37] T. Grenager, Klein, and C. Manning. Unsupervised learning of field segmentation models for information extraction. In *Proceedings of the Forty Third Annual Meeting of the Association for Computational Linguistics*, 2005.

[38] Aria Haghighi, John Blitzer, and Dan Klein. Better word alignments with supervised ITG models. In *Proceedings of the Forty Seventh Annual Meeting of the Association for Computational Linguistics*, 2009.

[39] Aria Haghighi and Dan Klein. Unsupervised coreference resolution in a nonparametric bayesian model. In *Proceedings of the Forty Fifth Annual Meeting of the Association for Computational Linguistics*, 2007.

[40] Zellig S. Harris. Distributional structure. In J. Katz, editor, *The Philosophy of Linguistics*, pages 26–47. Oxford University Press, New York, 1985.

[41] Marti Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, 1992.

[42] Geoff Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

[43] Liang Huang. *Forest-based Algorithms in Natural Language Processing.* PhD thesis, University of Pennsylvania, 2008.

[44] H. Kautz, B. Selman, and Y. Jiang. A general stochastic approach to solving problems with hard and soft constraints. In D. Gu, J. Du, and P. Pardalos, editors, *The Satisfiability Problem: Theory and Applications*, pages 573–586. American Mathematical Society, New York, NY, 1997.

[45] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N Ueda. Learning systems of concepts with an infinite relational model. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*, Boston, MA, 2006. AAAI Press.

[46] K. Kersting and L. De Raedt. Towards combining inductive logic programming with bayesian networks. In *Proceedings of the Eleventh International Conference on Inductive Logic Programming*, pages 118–131. Springer, 2001.

[47] Chloe Kiddon and Pedro Domingos. Leveraging ontologies for lifted probabilistic inference and learning. In *Proceedings of AAAI Workshop on Statistical Relational AI*, 2010.

[48] Pekka Kilpelainen. *Tree Matching Problems with Applications to Structured Text databases.* Ph.D. Thesis, Department of Computer Science, University of Helsinki, 1992.

[49] Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:180–82, 2003.

[50] Dan Klein and Christopher D. Manning. A* parsing: Fast exact viterbi parse selection. In *HLT-NAACL*, 2003.

[51] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the Forty First Annual Meeting of the Association for Computational Linguistics*, pages 423–430, 2003.

[52] Stanley Kok and Pedro Domingos. Learning the structure of Markov logic networks. In *Proceedings of the Twenty Second International Conference on Machine Learning*, pages 441–448, Bonn, Germany, 2005. ACM Press.

[53] Stanley Kok and Pedro Domingos. Statistical predicate invention. In *Proceedings of the Twenty Fourth International Conference on Machine Learning*, Corvallis, Oregon, 2007. ACM Press.

[54] Stanley Kok and Pedro Domingos. Extracting semantic networks from text via relational clustering. In *Proceedings of the Nineteenth European Conference on Machine Learning*, pages 624–639, Antwerp, Belgium, 2008. Springer.

[55] Stanley Kok and Pedro Domingos. Learning markov logic network structure via hypergraph lifting. In *Proceedings of the Twenty Sixth International Conference on Machine Learning*, pages 505–512, Montreal, Canada, 2009. Omnipress.

[56] Stanley Kok and Pedro Domingos. Learning markov logic networks using structural motifs. In *Proceedings of the Twenty Seventh International Conference on Machine Learning*, Haifa, Israel, 2010. Omnipress.

[57] Stanley Kok, Parag Singla, Matt Richardson, Pedro Domingos, Marc Sumner, Hoifung Poon, and Daniel Lowd. The alchemy system for statistical relational ai. Technical report, Dept. of CSE, Univ. of Washington, http://alchemy.cs.washington.edu/, 2009.

[58] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques.* MIT Press, 2009.

[59] Alex Kulesza and Fernando Pereira. Structured learning with approximate inference. In *Advances in Neural Information Processing Systems 20*, 2008.

[60] Nicholas Kushmerick. Wrapper induction: efficiency and expressiveness. *Artificial Intelligence*, 118:15–68, 2000.

[61] S. Lawrence, K. Bollacker, and C. L. Giles. Autonomous citation matching. In *Proceedings of International Conference on Autonomous Agents*, 1999.

[62] Honglak Lee, MarcAurelio Ranzato, Yoshua Bengio, Geoff Hinton, Yann LeCun, and Andrew Y. Ng, editors. *Deep Learning and Unsupervised Feature Learning Workshop NIPS 2010*, 2010.

[63] Percy Liang, Michael Jordan, and Dan Klein. Learning dependency-based compositional semantics. In *Proceedings of the Forty Ninth Annual Meeting of the Association for Computational Linguistics*, 2011.

[64] Percy Liang and Dan Klein. Analyzing the errors of unsupervised learning. In *Proceedings of the Forty Sixth Annual Meeting of the Association for Computational Linguistics*, pages 879–887, Columbus, OH, 2008. ACL.

[65] Dekang Lin. Automatic retrieval and clustering of similar words. In *COLING-ACL98*, 1998.

[66] Dekang Lin. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*, Granada, Spain, 1998. ELRA.

[67] Dekang Lin and Patrick Pantel. DIRT - discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–328, San Francisco, CA, 2001. ACM Press.

[68] Daniel Lowd and Pedro Domingos. Efficient weight learning for Markov logic networks. In *Proceedings of the Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 200–211, Warsaw, 2007. Springer.

[69] X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the Forty Second Annual Meeting of the Association for Computational Linguistics*, 2004.

[70] Alexander Maedche. *Ontology learning for the semantic Web.* Kluwer Academic Publishers, Boston, Massachusetts, 2002.

[71] M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330, 1993.

[72] Andrew McCallum, Ronald Rosenfeld, Tom Mitchell, and Andrew Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.

[73] Andrew McCallum and Ben Wellner. Conditional models of identity uncertainty with application to noun coreference. In *Advances in Neural Information Processing Systems 17*, 2005.

[74] Lilyana Mihalkova and Raymond Mooney. Bottom-up learning of markov logic network structure. In *Proceedings of the Twenty Fourth International Conference on Machine Learning*, pages 625–632, Corvallis, Oregon, 2007.

[75] Saif Mohammad, Bonnie Dorr, and Graeme Hirst. Computing word-pair antonymy. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 982–991, Honolulu, HI, 2008. ACL.

[76] Raymond J. Mooney. Learning for semantic parsing. In *Proceedings of the Eighth International Conference on Computational Linguistics and Intelligent Text Processing*, pages 311–324, Mexico City, Mexico, 2007. Springer.

[77] S. Muggleton. Predicate invention and utilisation. *Journal of Experimental and Theoretical Artificial Intelligence*, 6, 1994.

[78] V. Ng. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of the Forty Third Annual Meeting of the Association for Computational Linguistics*, 2005.

[79] Vincent Ng. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the Forty Eighth Annual Meeting of the Association for Computational Linguistics*, 2010.

[80] L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147–177, 1997.

[81] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, New York, NY, 2006.

[82] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *Advances in Neural Information Processing Systems 15*, 2003.

[83] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.

[84] Slav Petrov. *Coarse-to-Fine Natural Language Processing*. PhD thesis, Department of Computer Science, University of Berkeley, 2009.

[85] Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL*, 2007.

[86] Hoifung Poon, Colin Cherry, and Kristina Toutanova. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2009.

[87] Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*, pages 458–463, Boston, MA, 2006. AAAI Press.

[88] Hoifung Poon and Pedro Domingos. Joint inference in information extraction. In *Proceedings of the Twenty Second National Conference on Artificial Intelligence*, pages 913–918, Vancouver, Canada, 2007. AAAI Press.

[89] Hoifung Poon and Pedro Domingos. Joint unsupervised coreference resolution with Markov logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 649–658, Honolulu, HI, 2008. ACL.

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

[91] Hoifung Poon and Pedro Domingos. Machine reading: A "killer app" for statistical relational AI. In *Proceedings of the Statistical Relational AI Workshop at AAAI-10*, 2010.

[92] Hoifung Poon and Pedro Domingos. Unsupervised ontological induction from text. In *Proceedings of the Forty Eighth Annual Meeting of the Association for Computational Linguistics*, pages 296–305, Uppsala, Sweden, 2010. ACL.

[93] Matt Richardson and Pedro Domingos. Building large knowledge bases by mass collaboration. In *Proceedings of the Second International Conference on Knowledge Capture*, pages 129–137, Sanibel Island, FL, 2003. ACM Press.

[94] Matt Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[95] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82:273–302, 1996.

[96] Stefan Schoenmackers, Oren Etzioni, and Daniel S. Weld. Scaling textual inference to the web. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008.

[97] B. Selman, H. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, 1996.

[98] Parag Singla and Pedro Domingos. Discriminative training of Markov logic networks. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 2005.

[99] Parag Singla and Pedro Domingos. Memory-efficient inference in relational domains. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*, 2006.

[100] Parag Singla and Pedro Domingos. Markov logic in infinite domains. In *Proceedings of the Twenty Third Conference on Uncertainty in Artificial Intelligence*, pages 368–375, Vancouver, Canada, 2007. AUAI Press.

[101] Rion Snow, Daniel Jurafsky, and Andrew Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING/ACL 2006*, 2006.

[102] Mohammad S. Sorower, Thomas G. Dietterich, Janardhan Rao Doppa, Prasad Tadepalli, and Xiaoli Fern. Learning rules from incomplete examples via a probabilistic mention model. In *IJCAI Workshop on Learning by Reading and its Applications in Intelligent Question-Answering*, 2011.

[103] S. Staab and R. Studer. *Handbook on ontologies*. Springer, 2004.

[104] Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago - a large ontology from Wikipedia and WordNet. *Journal of Web Semantics*, 2008.

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

[106] Don R. Swanson and Neil R. Smalheiser. An interactive system for finding complementary literatures: a stimulus to scientific discovery. *Artificial Intelligence*, 91, 1997.

[107] Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, 2002.

[108] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of Human Language Technologies: The 2003 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2003.

[109] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. A bayesian model for unsupervised semantic parsing. In *Proceedings of the Forty Ninth Annual Meeting of the Association for Computational Linguistics*, 2011.

[110] Jun-ichi Tsujii. Thesaurus or logical ontology, which do we need for mining text? In *Proceedings of the Language Resources and Evaluation Conference*, 2004.

[111] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. A model-theoretic coreference scoring scheme. In *Proceedings of MUC-6*, 1995.

[112] W. Wei, J. Erenrich, and B. Selman. Towards efficient sampling: Exploiting random walk strategies. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 2004.

[113] M. Wellman, J. S. Breese, and R. P. Goldman. From knowledge bases to decision models. *Knowledge Engineering Review*, 7, 1992.

[114] Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, 2004.

[115] Dekai Wu, Pascale Fung, Marine Carpuat, Chikiu Lo, Yongsheng Yang, and Zhaojun Wu. Lexical semantics for statistical machine translation. In Joseph Olive, Caitlin Christianson, and John McCary, editors, *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*. Springer, 2010.

[116] Fei Wu and Daniel S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceedings of the Seventeenth International Conference on World Wide Web*, Beijing, China, 2008.

[117] Alexander Yates and Oren Etzioni. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296, 2009.

[118] J. M. Zelle, R. J. Mooney, and J. B. Konvisser. Combining top-down and bottom-up techniques in inductive logic programming. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 343–351, San Mateo, California, 1994. Morgan Kaufmann.

[119] Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammers. In *Proceedings of the Twenty First Conference on Uncertainty in Artificial Intelligence*, pages 658–666, Edinburgh, Scotland, 2005. AUAI Press.

[120] Luke S. 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*, pages 878–887, Prague, Czech, 2007. ACL.

VITA

Hoifung Poon grew up in Guangzhou, China. He earned a Bachelor of Science degree in Computer Science from Sun Yat-Sen University in 1998, a Master of Science in Computer Science and Engineering from the University of Washington in 2006, and a Doctor of Philosophy in Computer Science and Engineering from the University of Washington in 2011.