WEB-SCALE KNOWLEDGE-BASE CONSTRUCTION VIA STATISTICAL INFERENCE AND LEARNING

by

Feng Niu

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

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2012

Date of final oral examination: 08/14/12
This dissertation is approved by the following members of the Final Oral Committee:

Christopher Ré, Assistant Professor, Computer Sciences
AnHai Doan, Associate Professor, Computer Sciences
Jude W. Shavlik, Professor, Computer Sciences
Jeffrey F. Naughton, Professor, Computer Sciences
Mark Craven, Professor, Biostatistics and Medical Informatics

Acknowledgments

I owe my deepest gratitude to Christopher Ré. Chris possesses limitless energy, infectious enthusiasm, and an immense ability to impart his wisdom and grand visions. I am forever in debt for his trust and encouragement, and for the skills that I have learned from him. His relentless pursuit of elegance and clarity will forever be an inspiration to me.

I am profoundly grateful to my co-advisor AnHai Doan and my mentor Jude Shavlik. It was AnHai's research that planted a seed of intense curiosity in me – wild imaginations about futuristic information systems grew and persisted ever since. Jude has always been supportive and encouraging to my research; his meticulous care for details will stand as a shining beacon of rigor and depth in my intellectual endeavors. AnHai and Jude introduced me to the fascinating wonderlands of data management and machine learning that form the cornerstones of this dissertation.

It is an honor for me to have Jeff Naughton and Mark Craven on my committee. I am indebted to Shuchi Chawla, who provided supportive and patient mentorship to me when I was embarking graduate school. Her encouragement was invaluable in cultivating the confidence of a starting researcher. I would like to thank Venky Ganti, for taking me under his wings during my internship at Google. It is a pleasure to thank the students in the Hazy research group, the database group, and the Computer Sciences department. Their company made the last few years not only an intellectually vibrant journey, but a fun ride as well.

My research would not have been possible without the generous support from the DARPA Machine Reading Grant FA8750-09-C-0181 and Miron Livny's Condor research group. I am grateful to Michelle Craft and Kenneth Hahn for being guardian angels of our infrastructure.

Lastly, I would like to thank my parents, whose endless love and support are the ultimate driving force that placed me where I am.

Contents

Contents ii

List of Figures iv

List of Tables vi

List of Algorithms vii

Glossary viii

Abstract xi

1 Introduction 1
- 1.1 Problem Space and Challenges . . . 3
- 1.2 Technical Contributions . . . 6
- 1.3 Outline . . . 8

2 Preliminaries 9
- 2.1 Knowledge-base Construction . . . 9
- 2.2 Statistical Modeling . . . 14
- 2.3 Statistical Inference . . . 25
- 2.4 Statistical Learning . . . 35
- 2.5 Additional Related Work . . . 36

3 Knowledge-base Construction in Elementary 42
- 3.1 Conceptual Model and Architecture . . . 42
- 3.2 Examples . . . 46
- 3.3 Scaling Feature Extraction . . . 53
- 3.4 Effectiveness of Statistical Inference . . . 54

4 Scaling Distant Supervision for KBC 60

4.1 Motivations 60
4.2 Distant Supervision Methodology 63
4.3 Experiments 70
4.4 Discussion 82

5 Scaling Markov Logic using an RDBMS 83
5.1 Motivations, Challenges, and Contributions 83
5.2 Tuffy Systems 87
5.3 Experiments 95
5.4 Summary 103

6 Scaling Markov Logic via Task Decomposition 104
6.1 Motivations, Challenges, and Contributions 104
6.2 Dual Decomposition for MLNs 106
6.3 Specialized Tasks 111
6.4 Experiments 116
6.5 Summary 122

7 Conclusion and Future Work 124
7.1 Conclusion 124
7.2 Future Work 124

Bibliography 127

Appendix A Tuffy 145
A.1 Additional Systems Details 145
A.2 Additional Experiments 151
A.3 Additional Features in Tuffy 153

Appendix B Felix 155

List of Figures

1.1 Screen-shots from DeepDive 2

2.1 Illustrating mentions and entities 9
2.2 Correlation structures of logistic regression 15
2.3 Correlation tructure of conditional random fields 15
2.4 Markov random fields vs. factor graphs 18
2.5 Graphical model for correlation clustering 19
2.6 An example Markov logic program 21
2.7 A portion of the graphical model underlying an MLN 24

3.1 An illustration of the KBC model in Elementary. 43
3.2 Architecture of Elementary 44
3.3 GeoDeepDive's development pipeline 51
3.4 Screen-shot from GeoDeepDive 53
3.5 Comparing how different signals impact the quality of KBC . 58

4.1 Workflow of distant supervision 63
4.2 Relations used in our experiments on distant supervision . . . 66
4.3 Impact of input sizes under the TAC-KBP metric 72
4.4 Impact of input sizes under the MR-KBP metric 73
4.5 Impact of corpus size changes under the TAC-KBP metric . . . 74
4.6 Impact of human feedback amount under the TAC-KBP metric 75
4.7 TAC-KBP quality with human labels only 76
4.8 Impact of input sizes under the Freebase held-out metric . . . 77
4.9 Impact of input sizes with $\nu = 0.5$ under the TAC-KBP metric 78
4.10 Impact of input sizes with $\nu = 0.9$ under the TAC-KBP metric 78
4.11 Impact of input sizes with a 33%-down-sampled Freebase KB 79
4.12 Impact of corpus size on TAC-KBP with ClueWeb 80
4.13 Impact of corpus size on MR-KBP with ClueWeb 80

4.14 Impact of corpus size on individual relations 81

5.1 Comparison of architectures 89
5.2 A loosely connected graph for Example 5.3 93
5.3 Time-cost plots of Alchemy vs. Tuffy 97
5.4 Comparing Alchemy and different versions of Tuffy 99
5.5 Comparing Tuffy vs. Tuffy without partitioning 101
5.6 Effect of further partioning in Tuffy 102

6.1 A program-level decomposition for Example 6.1 108
6.2 Time-cost performance of Tuffy vs. Felix 119

A.1 Effect of partitioning on Example 5.1 149

List of Tables

3.1 Profiles of input signals for different KBC tasks 57

4.1 TAC-KBP F1 score of corner cases 73
4.2 Statistical significance of the Impact of corpus size changes . . 74
4.3 Statistical significance of the Impact of human feedback sizes . 75
4.4 Statistical significance tests of the impact of corpus size 81

5.1 Dataset statistics in the Tuffy experiments 96
5.2 Comparing Alchemy vs. Tuffy's grounding time 98
5.3 Comparing the WalkSAT speed of Alchemy vs. Tuffy 99
5.4 Comparing the space efficiency of Alchemy vs. Tuffy 99
5.5 Comparing Tuffy vs. Tuffy without partitioning 101

6.1 MRF-level vs. program-level decompositions 106
6.2 Example specialized tasks in Felix 112
6.3 Properties assigned to predicates by the Felix compiler 114
6.4 Tasks and their required properties in Felix 115
6.5 Dataset statistics for the Felix experiments 118
6.6 Decomposition schemes of each task in Felix 120
6.7 Performance and quality comparison on specialized tasks . . . 122

A.1 Grounding time in seconds 152
A.2 Comparison of execution time in seconds 152

B.1 Notation used in this section 155

List of Algorithms

1 Viterbi algorithm for CRFs 26
2 Forward-backward algorithm for CRFs 26
3 Ailon et al.'s correlation clustering algorithm [2] 27
4 Singh et al.'s correlation clustering algorithm [119] 28
5 SumProduct algorithm for marginal inference 29
6 Gibbs sampling for graphical models 30
7 Lazy grounding procedure in Alchemy 31
8 WalkSAT for MAP inference in Markov logic 31
9 SampleSAT 33
10 MC-SAT for marginal inference in Markov logic 34
11 MLN Grounding in SQL 145
12 A Simple MRF Partitioning Algorithm 150

GLOSSARY

Alchemy A reference implementation of Markov logic developed at University of Washington.

AncientText A demo project of Elementary where we help English professors organize 140K ancient books.

clique A subgraph where every pair of nodes are connected.

correlation clustering A clustering model in machine learning where the relationships (similarity and dis-similarity) between objects are known instead of the actual representation of the objects.

crowdsourcing An increasingly popular problem-solving process where tasks are distributed to an undefined group of people (as opposed to, e.g., employees).

dependency path A path in the graph (usually a tree) of tokens in a sentence where the (directed) edges represent dependency-grammar relations between tokens.

DeepDive A demo project of Elementary where we enhance Wikipedia with knowledge-base contruction over 500M web pages in the ClueWeb09 corpus.

domain knowledge Knowledge about the context or environment of an application (e.g., a KBC task).

dual decomposition A classic technique in mathematical optimization that approximates a difficult (constrained) optimization problem with a set of simpler problems.

Elementary Our prototype system for knowledge-base construction that encapsulates the techniques described in this dissertation.

F1 score The harmonic mean of precision and recall.

feature extraction The process of computing characteristic measurements (e.g., Boolean or numerical variables) over the input data of statistical inference or learning.

Felix Our prototype system for the task-decomposition approach to Markov logic inference.

GeoDeepDive A demo project of Elementary where we help geoscientists organize information in 20K journal papers in geology.

graphical model Probabilistic graphical model, namely a framework for succinctly representing probability distirbutions with a graph encoding conditional independency between random variables.

hypergraph A generalization of graph where each edge can connect multiple vertices.

knowledge base A database of facts or assertions.

Lagrange multiplier A classic technique in mathematical optimization to convert a constrainted problem in an unconstrainted problem; used by dual decomposition.

Markov logic A language that uses weighted first-order logic rules to represent Markov random fields with Boolean variables.

MLN Markov logic network, namely a Markov logic program.

MRF Markov random field, namely an undirected graphical model representing (pair-wise) correlations between random variables.

prcision The ratio between correct positive predictions and total positive predictions.

RDBMS A relational database management system.

recall The ratio between correct positive predictions and the sum of correct positive predictions and incorrect negative predictions.

statistical inference The process of making probabilistic predictions from data, often via mathematical optimization.

statistical learning The process of computing or estimating the best statistical model based on training examples (i.e., pairs of input data and desired predictions).

statistical significance A statistical assessment of whether measurements of an event reflect a pattern or just chance.

Tuffy Our prototype system for Markov logic inference using an RDBMS.

Abstract

Knowledge-base construction (KBC) is the process of populating a knowledge base (KB) with facts (or assertions) extracted from text. Bearing the promise of being a key technology of next-generation information systems, KBC has garnered tremendous interest from both academia and industry. A general trend in state-of-the-art KBC systems is the use of statistical inference and learning, which allow a KBC system to combine a wide range of data resources and techniques. In particular, two general techniques have gained significant interest from KBC researchers: the distant supervision technique for statistical learning, and the Markov logic framework for statistical inference.

This dissertation examines the application of distant supervision and Markov logic to web-scale KBC. Specifically, to fill a gap in the literature, we perform a systematic study on distant supervision to evaluate the impact of input sizes on the quality of KBC, hence providing guidelines for KBC system builders. While Markov logic has been shown to be effective for many text-understanding applications including KBC, the scalability of statistical inference in Markov logic remains a critical challenge. Inspired by ideas from data management and optimization, we propose two novel approaches that scale up Markov logic by orders of magnitude. Furthermore, we encapsulate our research findings into a general-purpose KBC system called Elementary, and deploy it to build a demonstration called DeepDive that reads hundreds of millions of web pages to enhance Wikipedia. Based on the above contributions, this dissertation shows that the distant supervision technique for statistical learning and the Markov logic framework for statistical inference are indeed effective approaches to web-scale KBC.

1 Introduction

Knowledge-base construction (KBC) is the process of populating a knowledge base (KB) with facts (or assertions) extracted from text. It has recently received tremendous interest from academia, e.g., KnowItAll [33], DBLife [28], NELL [17], YAGO [57, 84], and from industry, e.g., IBM's DeepQA [39], Microsoft's EntityCube [149], and Google's Knowledge Graph.[1] To achieve high quality, these systems leverage a wide variety of data resources and KBC techniques. A crucial challenge that these systems face is coping with imperfect or conflicting information from multiple sources [140]. To address this challenge, a growing trend is to use machine learning and statistical inference.

In particular, two general techniques have gained significant interest from KBC researchers: the *distant supervision* technique for statistical learning [17, 51, 84, 141, 144, 149], and the *Markov logic* framework for statistical inference [84, 110, 126, 133, 149]. Toward the goal of applying these approaches to web-scale KBC, this dissertation examines several key scalability issues for these techniques. The studies presented in this dissertation demonstrate that the distant supervision technique for statistical learning and the Markov logic framework for statistical inference are indeed effective approaches to web-scale KBC. Below is a summary of our main contributions:

1. **Distant Supervision for Statistical Learning.** To fill a gap in the literature, we have performed a systematic study [148] to empirically validate the effectiveness of scaling the amount of input data resources for *distant supervision* [82], an increasingly popular machine-learning technique for KBC. Our results shed light on how different input data resources contribute to the quality of KBC.

[1] `http://www.google.com/insidesearch/features/search/knowledge.html`

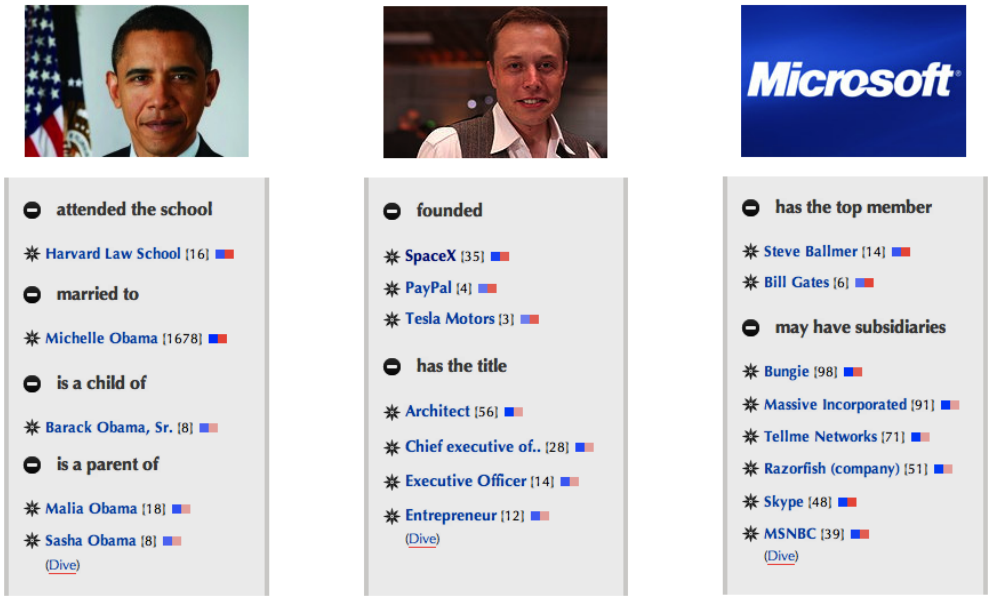


Figure 1.1: *Sample facts about Barack Obama, Elon Musk, and Microsoft that are extracted by* DeepDive. *The numbers in brackets indicate how many times a fact is mentioned. The blue and red color tags indicate frequency and recency.*

2. **Markov Logic for Statistical Inference.** A key technical challenge for applying statistical methods to KBC is scalability [140]. Inspired by ideas from data management and optimization, we have designed and implemented several novel approaches to statistical inference in *Markov logic* [103], a statistical-inference framework that has been used in many text-understanding applications (including KBC). These approaches achieve orders of magnitude higher efficiency than prior approaches [87, 88].

3. **Demonstration.** Based on the above research findings, we have developed an end-to-end domain-independent KBC platform called Elementary [90]. Using Elementary, we have built a web-scale KBC system called DeepDive[2] that performs deep linguistic analysis and sophisticated

[2] `http://hazy.cs.wisc.edu/deepdive`

statistical processing over hundreds of millions of web pages to enrich Wikipedia[3] (see Figure 1.1). In addition, we have applied ELEMENTARY to build KBC systems over a 140K-ancient-book corpus for English professors, and over a collection of 20K journal papers for geoscientists.

1.1 Problem Space and Challenges

The ultimate goal of this dissertation is to build a web-scale knowledge-base construction system using statistical inference and learning. We believe that we have accomplished this goal as the techniques in this dissertation and the DEEPDIVE demonstration [89] support our central thesis: the distant supervision technique for statistical learning and the Markov logic framework for statistical inference are indeed effective approaches to web-scale KBC. We briefly describe the problem space of this dissertation. (Further discussion of related work can be found in Chapter 2.)

Knowledge-base construction is closely related to *information extraction*, *knowledge harvesting*, and *machine reading*, all of which strive to devise algorithms or build systems that help machines (or human analysts) "make sense of" data without explicit semantics; e.g., natural-language text and HTML tables. An example KBC task is to populate a biographical database (covering, e.g., people's names, professions, affiliations, etc.) from news and blog articles. The main focus of this dissertation is *building KBC systems* rather than inventing or improving individual KBC algorithms: the goal is to build a KBC system that is able to *integrate* diverse data resources and algorithms (Chapter 3). (Although the techniques in this dissertation can be potentially applied to the emerging field of *open information extraction* [34, 141, 146], we exclusively focus on the scenarios where the target knowledge base has a specified schema.)

Indeed, to construct high-quality knowledge bases from text, researchers

[3] `http://en.wikipedia.org`

have considered a wide range of data resources and techniques (Chapter 2); e.g., pattern matching with dictionaries listing entity names [107], bootstrapping from existing knowledge bases like Freebase[4] and YAGO [125], disambiguation using web links and search results [31, 48], rule-based extraction with regular expressions curated by domain experts [21, 28], training statistical models with annotated text [70], etc. All these resources are valuable because they are complementary in terms of cost, quality, and coverage; ideally one would like to be able to use them all. To take advantage of different kinds of data resources, a major problem that KBC systems face is coping with imperfect or conflicting information from multiple sources [140]. (We use the term "information" to refer to both data and algorithms that can be used for a KBC task.) To address this issue, several recent KBC projects [17, 57, 71, 84, 149] use statistical inference and learning to combine different data resources.

In particular, two generic techniques have gained significant interest in the KBC community: (1) the *distant supervision* technique [82] that addresses the scarcity issue of training data when applying statistical learning to KBC, and (2) the *Markov logic* framework for statistical inference [103] that is able to incorporate diverse signals such as those listed above. Sample KBC systems employing either or both techniques include NELL [17], SOFIE [126], Prospera [84], KnowItAll [33], Snowball [1], StatSnowball/EntityCube [149], LUCHS [51], and Kylin [141, 144]. On the other hand, the generality of these two techniques makes it possible to build a general KBC system with these techniques at the core. Following this trend, this dissertation addresses the scalability issues of these two techniques:

Scaling Distant Supervision To handle the sheer variations of textual expressions to refer to an entity or mention a relationship, KBC systems often use statistical learning. Statistical learning, however, requires train-

[4] `http://freebase.com`

ing examples such as manual annotations that are scarce resources. To address this challenge, an increasingly popular technique is *distant supervision* [82], where one mitigates such scarcity by generating lower-quality but high-volume training data from scalable data sources such as text corpora, existing knowledge bases, and human feedback via crowdsourcing.

Although the distant supervision technique has been shown to be effective in many KBC tasks [17, 84, 149], it was unclear how the amount of input *quantitatively* impacts the quality of KBC. To fill this gap and provide guidelines for KBC, we perform a detailed quantitative study on distant supervision (Chapter 4).

Scaling Markov Logic To integrate diverse signals and techniques for KBC (including the statistical models learned via distant supervision), researchers have applied a statistical inference framework called *Markov logic* [103] to a wide range of text-related applications. In Markov logic, one can write first-order logic rules with weights (that intuitively model our confidence in a rule); this allows one to capture rules that are likely, but not certain, to be correct. When trying to apply Markov logic to KBC, we found that state-of-the-art Markov logic systems such as Alchemy[5] do not scale to our datasets.

To address this issue, state-of-the-art KBC systems that employ Markov logic typically implement inference algorithms customized for the particular Markov logic programs that they use [84, 110, 126, 133, 149]. The downside is that they are not able to support rules or correlations that their specialized inference algorithms cannot handle. However, extensive experiments have shown that such rules or correlations can significantly improve the quality of KBC [90, 97, 98, 99, 120]. As a result, we design, implement, and evaluate two novel but orthogonal approaches that scale up general Markov logic inference by leveraging data management and op-

[5] `http://alchemy.cs.washington.edu`

timization techniques. These approaches are embodied in our open-source TUFFY[6] (Chapter 5) and FELIX[7] (Chapter 6) systems.

1.2 Technical Contributions

The technical contributions of this dissertation can be divided into four parts: the ELEMENTARY architecture to knowledge-base construction, a systematic study of distant supervision, and two novel techniques for scaling Markov logic inference. These contributions collectively support our thesis that the distant supervision technique for statistical learning and the Markov logic framework for statistical inference are indeed effective approaches to web-scale KBC. We briefly describe each of them below.

Elementary Architecture to KBC ELEMENTARY is our prototype KBC system that is able to combine diverse data resources and different KBC techniques via machine learning and statistical inference to construct knowledge bases [90]. Using Markov logic, ELEMENTARY accepts both domain-knowledge rules and classical machine-learning models such as conditional random fields, thereby integrating different data resources and KBC techniques in a principled manner. Using ELEMENTARY, we have implemented a solution to the TAC-KBP challenge[8] with quality comparable to the state of the art, as well as a demonstration system called DEEPDIVE [89] that automatically and continuously enriches Wikipedia with structured data by reading millions of web pages on a daily basis. To quantitatively validate the effectiveness of ELEMENTARY's approach to KBC, we experimentally show that its ability to incorporate diverse signals has positive impacts on the quality of many KBC tasks.

[6] http://hazy.cs.wisc.edu/tuffy
[7] http://hazy.cs.wisc.edu/felix
[8] http://nlp.cs.qc.cuny.edu/kbp/2010/

Quantitative Study of Distant Supervision Distant supervision is an increasingly popular technique in KBC. There had been, however, no study evaluating how the scale of input data impacts the quality of distant supervision. To fill this gap, we empirically assess the impact of big data versus the crowd: we use up to 100M documents and tens of thousands of crowd-sourced human labels. We find that, while increasing the corpus size consistently and significantly improves quality (precision and recall), human feedback has a significant impact only when accompanied by a large corpus [148]. We observe that distant supervision is often *recall gated*. Thus, a rule of thumb for developing distant supervision-based KBC systems is that one should prioritize maximizing the diversity of features and the coverage of training corpus.

Scaling Markov Logic using an RDBMS Tuffy is based on the observation that MLN inference consists of a grounding step that essentially performs relational operations, and a search (or sampling) step that often comprises multiple independent subproblems [87]. In contrast to Alchemy that essentially performs grounding with hard-coded nested-loop joins, Tuffy translates grounding into SQL statements and uses a relational database management system (RDBMS) to execute them. This enables Tuffy to achieve orders of magnitude speed-up compared to Alchemy. The grounding step results in a graphical model called a *Markov random field* (MRF) that is the input to the search step. The MRF can be orders of magnitude larger than the input (evidence) database, and so may not fit in memory. To scale up the search step, Tuffy decomposes the MRF into multiple partitions, and solves each partition (in memory) in turn. A serendipitous consequence of such partitioning is that the search efficiency and result quality may improve substantially. We empirically validate this phenomenon and theoretically quantify such improvement.

Scaling Markov Logic via Task Decomposition FELIX is based on the observation that an MLN program (especially those for KBC) often contains routine subtasks such as classification and coreference resolution; these subtasks have specialized algorithms with high efficiency and quality. Thus, instead of solving a whole MLN with generic inference algorithms (which we call *monolithic* approaches), we could split the program into multiple parts and solve subtasks with corresponding specialized algorithms [88]. However, because different subtasks may share the same relation, there may be conflicts between predictions from different subtasks. To address this challenge, FELIX employs the classic *dual decomposition* technique that resolves conflicts by passing messages between subtasks. We experimentally validate that, on simple MLNs, FELIX achieves similar performance as monolithic inference approaches (e.g., TUFFY and ALCHEMY); on complex MLNs, FELIX substantially outperforms monolithic inference.

1.3 Outline

The remainder of this dissertation is organized as follows. In Chapter 2, we cover background material and related work on KBC as well as on statistical inference and learning. In Chapter 3, we describe the conceptual model and architecture of ELEMENTARY, as well as the effectiveness of incorporating diverse signals into a KBC task. In Chapter 4, we present a systematic and quantitative study on the distant supervision technique for KBC. In Chapters 5 and 6, we describe how we scale Markov logic using an RDBMS and a task-decomposition approach, respectively. We conclude in Chapter 7.

2 Preliminaries

We first describe three pieces of critical background material for this dissertation: (1) what signals (including data resources and techniques) are commonly used in KBC, (2) how these signals can be combined via statistical modeling, and (3) how to perform inference and learning on these statistical models. In Section 2.5, we describe further related work.

2.1 Knowledge-base Construction

We describe several common data resources and techniques for KBC.

2.1.1 Data Resources

We briefly recall two common types of data resources that are valuable for achieving high quality in KBC: mention-level data that deal with the structure in text, and entity-level data that deal with the structure in the target knowledge base (e.g., typing of relation arguments and constraints among relations). The distinction between mentions and entities is illustrated in Figure 2.1.

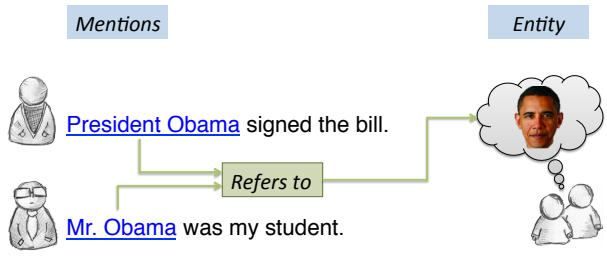


Figure 2.1: *Illustrating mentions in text of the entity Barack Obama.*

Mention-level Data *Mention-level data* are textual data with *mentions* of entities, relationships, events, etc., or linguistic resources modeling

a (natural) language itself. Mentions in a natural language are the raw data from which KBC data is primarily extracted. They provide examples of the variety of ways humans, writing for other humans, encode knowledge. They serve as the media of information to be extracted as well as examples of how information is encoded. Mention-level data are usually the primary input to a KBC task; they are often in the form of raw text corpora and sometimes also include annotated text. One could use annotated text to train statistical information extraction models (e.g., CRFs [70]). There are also unsupervised methods to make use of less structured mention-level data: a natural-language text corpus could be used to build statistical language models [73]; one could derive from anchor texts (of Wikipedia pages or web pages) how frequently an expression is used to refer to an entity and use this information to help disambiguate entity mentions [48]. *Mention-level models* are intermediate summaries of the structure in mention-level data. Examples include part-of-speech tags and parse trees of the input text, co-occurrence statistics of noun phrases and verb phrases [73], linguistic patterns or regular expressions for specific relations (e.g., that "X was born in Y" expresses the relation `BirthPlace` (X,Y)), and generic linguistic resources such as synonyms from WordNet [44, 92, 124].

Entity-level Data *Entity-level data* are relational data over the domain of conceptual entities (as opposed to language-dependent mentions). They include what are usually called knowledge bases, ontologies, and gazetteers (e.g., YAGO [84] and Freebase[1]). On the one hand, entity-level data are typical output of KBC systems. On the other hand, existing entity-level data resources are also valuable input for building KBC systems. For example, one could use comprehensive ontologies like YAGO to extract mentions of entities such as people and organizations with pattern matching. One

[1] `http://freebase.com`

could also use an existing knowledge base to type-check extraction results [126]; e.g., the second argument of the relation `EmployedBy` must be an organization. *Entity-level models* refer to (first-order) logical statements that encode common-sense (or domain-expertise) constraints and rules over relations of interest. For example, rules like "marriage is symmetric" and "a person's birth date is unique" intuitively would help improve biographical relation extraction. Indeed, such high-level constraints have been shown to have significant impacts on KBC quality [84].

2.1.2 KBC Techniques

We discuss several common techniques to make use of the above data resources for KBC: classical rule-based approaches, classical machine learning approaches, distant supervision, statistical inference, and human feedback.

Classical Rule-based Approaches Classical rule-based approaches to KBC (e.g., DBLife [28, 116] and SystemT [21, 79]) leverages the observation that, for many KBC tasks, there are often a small number of rules that can achieve both high precision and reasonable recall. Furthermore, because rules tend to be intuitive and rule-based systems tend to be efficient and scalable, rule-based approaches have the advantages of being accessible to many developers and easy to debug. For example, to identify mentions of people in the database research community, DBLife performs string matching between the corpus and heuristic variations of a dictionary of canonical person names (e.g., abbreviations and first/last name ordering). Because these heuristics cover most commonly used variation types, and because person-name ambiguities are rare, DBLife is able to achieve both high recall and high precision in this subtask. Furthermore, the first-order nature of such rules (e.g., one rule can apply to a dictionary of tens of thousands of names) means that developing rule-based KBC systems can

be very efficient. Another reason for the effectiveness of rule-based KBC is that the input to many KBC tasks are semi-structured text (e.g., HTML). Such semi-structured documents tend to share a handful of structures (e.g., if they originate from a small number of websites). Therefore, one could develop a few extraction rules to handle a large number of input documents. The development process of rule-based KBC systems is increasingly being assisted by statistical techniques [7, 68, 69, 79, 79, 83, 109].

Classical Machine Learning Approaches Classical machine learning approaches [19, 22, 70, 82], on the other hand, target KBC tasks that cannot be reasonably covered by a small set of deterministic rules. For example, to extract `HasSpouse` relationships from English text, one would be hard pressed to enumerate a set of possible expressions with high precision and high recall. To address this issue, classical statistical approaches employ machine learning models such as logistic regression and conditional random fields [70, 130] to learn model parameters from training examples, e.g., annotations of sentences that mention a `HasSpouse` relationship. A trend in classical statistical KBC approaches is to incorporate high-level knowledge (rules) such as "similar text spans are more likely to have the same label" [40, 129].

Distant Supervision High-quality annotation data are usually scarce and expensive to obtain. *Distant supervision* is an increasingly popular technique to use entity-level data to generate (silver-standard) annotations from mention-level data [15, 23, 46, 82]. The idea is to use the facts in an existing knowledge base to heuristically identify fact mentions in a text corpus via entity-mention mapping. For example, to populate a relation about people's birth places, one may start with a small set of tuples such as `BirthPlace`(Obama, Hawaii). Then one could consider sentences mentioning both "Obama" and "Hawaii" (e.g., "Obama was born in Hawaii") as positive examples. Textual patterns like "was born in" would then emerge

as being indicative for the `BirthPlace` relation and be used to extract more facts. The significance of this technique is that we can get large amounts of training data without human input. While distant supervision often generates noisy examples, the hope is that machine learning techniques (e.g., ℓ_1-regularization [149]) as well as redundancies in the training data help reduce such noisiness and select truly indicative patterns.

Distant supervision is widely used in recent KBC systems; e.g., NELL [17], Prospera [84], KnowItAll [33], Snowball [1], StatSnowball/EntityCube [149], LUCHS [51], and Kylin [141, 144]. To the best of our knowledge, Craven and Kumlien [23] were the first to apply distant supervision in a statistical learning-based approach. To improve the quality of distant supervision, there is recent work that incorporates probabilistic rules to the trained statistical model so as to counter the noises in distantly supervised examples [50, 106, 147].

Statistical Inference A recent trend of KBC is to integrate domain knowledge (e.g., in the form of heuristic rules or constraints) into statistical modeling and then perform statistical inference; e.g., NELL [17, 71] that tries to maintain consistency between several different extraction components, SOPHIE/Prospera [84, 126, 133] that incorporate entity typing and common-sense constraints, and StatSnowball/EntityCube [149] that enforces cross-relation constraints such as "`IsHusband` and `IsWife` are symmetric." A particularly popular statistical inference framework is Markov logic [84, 126, 133, 149]. To address the scalability challenges of Markov logic inference, these systems implement specialized inference algorithms customized for the particular MLN programs that they use. In contrast, Elementary scales Markov logic inference in general instead of for particular MLN programs. We validate the advantage of this approach in Section 3.4.

Human Feedback Developing KBC systems should be a continuous process – a KBC system is built and deployed, and then iteratively improved as more and more human feedback is integrated. Human feedback may come from different sources, e.g., developers who can spot and correct systematic errors, end-users who can provide low-level (e.g., sentence-level) feedback, and workers on crowdsourcing platforms who can generate large volumes of annotations. We can directly incorporate such human feedback to improve KBC result quality. Furthermore, we may use human feedback as additional training data (e.g., on top of distant-supervision training data) to improve the KBC system itself. An example KBC system that automatically incorporates user feedback is DBLife [18].

2.2 Statistical Modeling

Let X be a set of variables whose values are observed (i.e., *evidence*), and Y a set of random variables that are possibly related to X and to each other. For simplicity, we assume that both X and Y consist of only Boolean variables. It is straightforward to generalize the following definitions to all discrete variables. A *probability distribution* over a set of Boolean variables, say Y, is a function $p : \{0,1\}^{|Y|} \mapsto \mathbb{R}_+$ such that $\sum_{\mathbf{y} \in \{0,1\}^{|Y|}} p(\mathbf{y}) = 1$. Denote by $P(Y)$ the set of all possible probability distributions over Y, then a *statistical model* $\mathcal{M} : \{0,1\}^{|X|} \mapsto P(Y)$ is a function that maps each possible evidence state $\mathbf{x} \in \{0,1\}^{|X|}$ to a distribution over Y.[2] Depending on how specialized or generic a statistical model is and how it is represented, there are many kinds of statistical models. We describe several types of statistical models that are most related to this dissertation. We start with specialized statistical models characterized by the correlation structure over X and Y, then describe general graphical models, and lastly discuss Markov logic that use first-order representations for the correlation structure.

[2]Note to experts: we focus on discriminative and undirected models.

2.2.1 Specialized Statistical Models

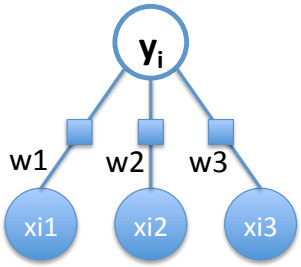


Figure 2.2: *Correlation structures of logistic regression. Shaded circles are evidence variables, hollow circles are variables to be predicted, and shaded squares indicate correlations between variables. Each square is associated with a function. Denote by* s *the sum of all squares' function values given an assignment* $\mathbf{y}$*, then the probability* $\Pr[Y = \mathbf{y}] \propto \exp\{s\}$.

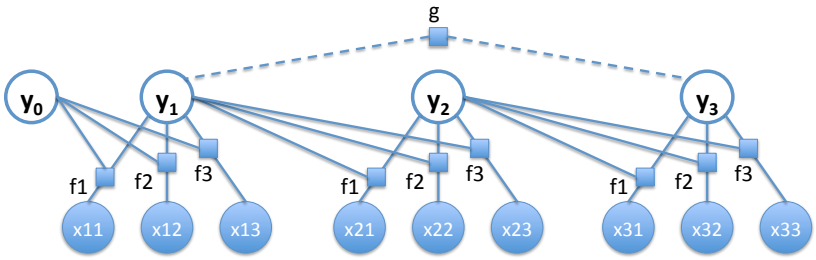


Figure 2.3: *Correlation structure of conditional random fields. Without (resp. with)* g *and the dashed links, this model is a linear-chain (resp. skip-chain) CRF.*

Logistic Regression A *logistic regression* model [47] assumes that each random variable $y_i \in Y$ is independent of each other and that each Y_i is related to exactly k evidence variables $\mathbf{x}_i = (x_{i1}, \ldots, x_{ik})$. More precisely, a logistic regression model is determined by k parameters $\mathbf{w} \equiv (w_1, \ldots, w_k) \in \mathbb{R}^k$:

$$\Pr[Y_i = 0] = \frac{1}{1 + \exp\{\mathbf{w} \cdot \mathbf{x}_i\}}; \qquad \Pr[Y_i = 1] = \frac{\exp\{\mathbf{w} \cdot \mathbf{x}_i\}}{1 + \exp\{\mathbf{w} \cdot \mathbf{x}_i\}}.$$

Intuitively, x_{ij} has an influence on y_i only if $x_{ij} = 1$ (sometimes called "activated"); the higher the magnitude of w_j, the more influence x_{ij} has on y_i. Moreover, positive w_j's "vote" toward $y_i = 1$ whereas negative w_j's "vote" toward $y_i = 0$; in particular, $\mathbf{w} = \mathbf{0}$ models the uniform distribution. The joint distribution is simply a product:

$$\Pr[Y = \mathbf{y}] = \prod_{1 \leqslant i \leqslant |Y|} \Pr[Y_i = y_i]$$

Figure 2.2 illustrates the correlation structure of logistic regression. The advantage of logistic regression models is their simplicity, but the downside is that it is not able to model correlations over Y.

Conditional Random Fields Conditional random fields (CRF) [70] extend logistic regression by also modeling correlations over Y. We describe the most commonly used CRF models: linear-chain CRFs. Linear-chain CRFs assume that the correlation structure over Y forms one or more sequences. Different sequences are independent to each other; for simplicity, we focus on one sequence. More precisely, let $Y = \{Y_1, \ldots, Y_n\}$, then there are direct correlations between Y_i and Y_{i+1} for $1 \leqslant i \leqslant n - 1$: Y_i not only depends on $\mathbf{x}_i$ but also depends on the values of Y_{i-1} and Y_{i+1}. Moreover, such dependency may not be as simple as the weight vector $\mathbf{w}$ in logistic regression; instead, we model such dependency using a *factor function* $f : \{0,1\}^{k+2} \mapsto \mathbb{R}$ of the form $f(\mathbf{x}_i, y_{i-1}, y_i)$. Assuming dummy value $y_0 = 0$, then the joint distribution is defined as

$$\Pr[Y = \mathbf{y}] = \frac{1}{Z} \exp\left\{\sum_{i=1}^{n} f(\mathbf{x}_i, y_{i-1}, y_i)\right\},$$

where

$$Z = \sum_{\mathbf{y} \in \{0,1\}^n} \exp\left\{\sum_{i=1}^{n} f(\mathbf{x}_i, y_{i-1}, y_i)\right\}.$$

In general, the factor function f could be further decomposed into a weighted sum to model signals that are intuitively independent, e.g.,

$$f(\mathbf{x}_i, y_{i-1}, y_i) = \sum_{j=1}^{k} \lambda_j f_j(x_{ij}, y_{i-1}, y_i),$$

where f_j might be Boolean functions and $\{\lambda_j \in \mathbb{R}\}_{j=1}^{k}$ comprise the parameters of this CRF.

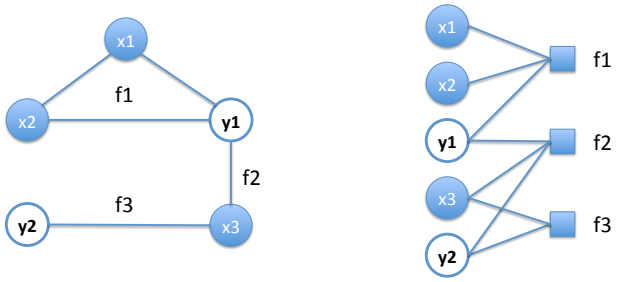
Such CRF models are frequently used to label sequential data such as text; e.g., labeling each token in a citation as AUTHOR or TITLE. There, two neighboring tokens intuitively have correlations such as "if neither token is a punctuation, then they are likely of the same label." Extending this idea further, one may also introduce correlations between non-neighboring tokens. For example, the *skip-chain* CRF model [129] extends linear-chain CRFs with additional factor functions between *distant* nodes; e.g., one may add a factor function $g(y_i, y_j)$ between tokens i, j if they have the same word (i.e., $x_{i1} = x_{j1}$ assuming $x_{\cdot 1}$ indicates the word of a token) to encourage $y_i = y_j$. To incorporate g, we redefine

$$\Pr[Y = \mathbf{y}] = \frac{1}{Z} \exp\left\{ \sum_{k=1}^{n} f(\mathbf{x}_k, y_{k-1}, y_k) + \sum_{i,j:x_{i1}=x_{j1}} g(y_i, y_j) \right\},$$

where Z is redefined accordingly. An example of g is $g(y_i, y_j) = 1 - |y_i - y_j|$. Figure 2.3 illustrates the correlation structures of linear-chain and skip-chain CRF models.

2.2.2 Graphical Models

We can generalize LR and CRF to a broad class of probabilistic distributions called *graphical models* [63, 130, 137]. The intuition is that, extending the correlation structures in Figure 2.3, we can actually allow arbitrary correlation factors (i.e., the shaded squares) over the variables. Formally,



(a) MRF (b) Factor graph

Figure 2.4: *Alternative representations of the same graphical model.*

denote by $\mathcal{P}(V)$ the power set of a set V. Let $F \subseteq \mathcal{P}(X \cup Y)$ be a set of *factors*; each factor $A \in F$ is associated with a function $f_A : \{0,1\}^A \mapsto \mathbb{R}$. The probability distribution is then defined by

$$\Pr[Y = \mathbf{y}] = \frac{1}{Z} \exp\left\{ \sum_{A \in F} f_A(\mathbf{x}_A, \mathbf{y}_A) \right\},$$

where

$$Z = \sum_{\mathbf{y} \in \{0,1\}^Y} \exp\left\{ \sum_{A \in F} f_A(\mathbf{x}_A, \mathbf{y}_A) \right\}.$$

Depending on what one wants to emphasize in an (undirected) graphical model, there are several different views:

1. To emphasize the dependency relationships between variables, Markov random fields (MRF) [63] represent a graphical model using a graph with nodes representing variables and cliques representing factors (see Figure 2.4(a)).

2. To explicitly show the correlation structure, *factor graphs* [67] represent a graphical model as a bipartite graph between variables and factors (see Figure 2.4(b)).

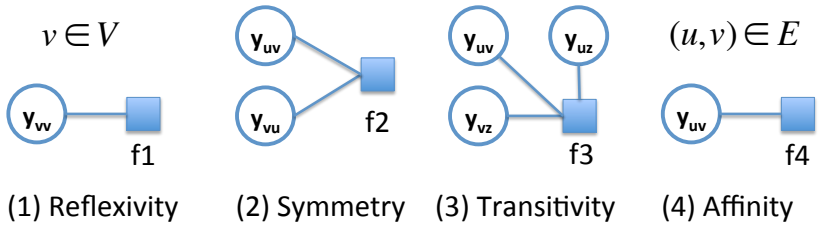


Figure 2.5: *A graphical model representing correlation clustering.*

3. To emphasize the fact that one could consider the distribution over Y as being conditional on X, conditional random fields (CRF) [130] tend to emphasize the distinction between X and Y (see Figure 2.3).

To demonstrate the expressiveness of graphical models, let us consider an example.

Example 2.1. *Correlation clustering (CC) [9, 27] is a common task for coreference resolution (coref) [98, 119], e.g., given a set of strings (say phrases in a document) we want to decide which strings represent the same real-world entity. The input to a CC task is a graph* $G = (V, E)$ *with edge weights* $w : E \mapsto \mathbb{R}$*; the output is a partitioning* $V = V_1 \cup \cdots \cup V_k$ *for some integer* $1 \leqslant k \leqslant |V|$*. Equivalently, we can represent such a partitioning with an equivalence relation* $R \subseteq V \times V$ *that satisfies the reflexivity, symmetry, and transitivity properties; i.e.,* $R(x, y)$ *iff nodes* x *and* y *belong to the same partition. Each valid* R *incurs a cost (called* disagreement cost*)*

$$\text{cost}_{\text{coref}}(R) = \sum_{\substack{x,y:(x,y)\notin R \\ \text{and } w((x,y))>0}} |w((x,y))| + \sum_{\substack{x,y:(x,y)\in R \\ \text{and } w((x,y))<0}} |w((x,y))|.$$

The objective is to find an R *with the minimum cost. To model CC using graphical models, let evidence variables* $X = \emptyset$*, prediction variables* $Y = \{y_{uv} | (u, v) \in V \times V\}$ *s.t.* $y_{uv} = 1$ *iff* $(u, v) \in R$*. There are four types of factors (see Figure 2.5):*

1. ***Reflexivity**: for each $y_{vv} \in \{y_{vv} \mid v \in V\}$, $f_1(y_{vv}) = 0$ if $y_{vv} = 1$ and $-\infty$ otherwise.*

2. ***Symmetry**: for each y_{uv}, $f_2(y_{uv}, y_{vu}) = 0$ if $y_{uv} = y_{vu}$ and $-\infty$ otherwise.*

3. ***Transitivity**: for each y_{uv} and y_{vz}, $f_3(y_{uv}, y_{vz}, y_{uz}) = -\infty$ if $y_{uv} = y_{vz} = 1$ but $y_{uz} = 0$, and 0 otherwise.*

4. ***Affinity**: for each $e = (u, v) \in E$, $f_4(y_{uv}) = -|w(e)|$ if $w(e)(2y_{uv}-1) < 0$ and 0 otherwise.*

Then one can see that the probability of an invalid partitioning R is 0 and the probability of each valid partitioning R is

$$\frac{1}{Z}\exp\{-\text{cost}_{\text{coref}}(R)\} \quad \textit{where} \quad Z = \sum_{R:R \textit{ is valid}} \exp\{-\text{cost}_{\text{coref}}(R)\}.$$

Thus, solving CC is equivalent to finding the most likely state of Y.

2.2.3 Markov Logic

From the above LR, CRF, and CC examples, we see that the factors in a graphical model can often be grouped into a small number of templates. Each of these specialized models is characterized by its templates: e.g., a linear-chain CRF model is defined by unigram and bigram factors, and a CC model is defined by the four types of factors in Figure 2.5. Thus, a valuable tool for statistical modeling is a framework to succinctly specify such templates in general. There are several such frameworks, e.g., Factorie [78] uses Java and Markov logic [30, 87, 103] uses weighted first-order logic rules. We focus on Markov logic, which is a core component of Elementary. We briefly describe Markov logic using an example that extracts affiliations between people and organizations from web text.

Schema	Evidence
pSimHard(per1, per2)	coOccurs('Ullman', 'Stanford Univ.')
pSimSoft(per1, per2)	coOccurs('Jeff Ullman', 'Stanford')
oSimHard(org1, org2)	coOccurs('Gray', 'San Jose Lab')
pSimSoft(org1, org2)	coOccurs('J. Gray', 'IBM San Jose')
coOccurs(per, org)	coOccurs('Mike', 'UC-Berkeley')
homepage(per, page)	coOccurs('Mike', 'UCB')
oMention(page, org)	coOccurs('Joe', 'UCB')
faculty(org, per)	faculty('MIT', 'Chomsky')
*affil(per, org)	homepage('Joe', 'Doc201')
*oCoref(org1, org2)	oMention('Doc201', 'IBM')
*pCoref(per1, per2)	...

Rules

weight	rule	
$+\infty$	$\texttt{pCoref}(p, p)$	(F_1)
$+\infty$	$\texttt{pCoref}(p1, p2) \Rightarrow \texttt{pCoref}(p2, p1)$	(F_2)
$+\infty$	$\texttt{pCoref}(x, y), \texttt{pCoref}(y, z) \Rightarrow \texttt{pCoref}(x, z)$	(F_3)
6	$\texttt{pSimHard}(p1, p2) \Rightarrow \texttt{pCoref}(p1, p2)$	(F_4)
2	$\texttt{pSimSoft}(p1, p2) \Rightarrow \texttt{pCoref}(p1, p2)$	(F_5)
$+\infty$	$\texttt{faculty}(o, p) \Rightarrow \texttt{affil}(p, o)$	(F_6)
8	$\texttt{homepage}(p, d), \texttt{oMention}(d, o) \Rightarrow \texttt{affil}(p, o)$	(F_7)
3	$\texttt{coOccurs}(p, o1), \texttt{oCoref}(o1, o2) \Rightarrow \texttt{affil}(p, o2)$	(F_8)
4	$\texttt{coOccurs}(p1, o), \texttt{pCoref}(p1, p2) \Rightarrow \texttt{affil}(p2, o)$	(F_9)
...		

Figure 2.6: *An example* MLN *program that performs three tasks jointly: 1. discover affiliation relationships between people and organizations* (affil)*; 2. resolve coreference among people mentions* (pCoref)*; and 3. resolve coreference among organization mentions* (oCoref)*. The remaining eight relations are evidence relations. In particular,* coOccurs *stores person-organization co-occurrences;* *Sim* *relations are string similarities.*

Syntax A Markov logic program (aka Markov logic networks, or MLN) consists of three parts: *schema*, *evidence*, and *rules* (see Figure 2.6). To tell Elementary what data will be provided or generated, the user provides a *schema*. Some relations are standard database relations, and we call these

relations *evidence*. Intuitively, evidence relations contain tuples that we assume are correct. In the schema of Figure 2.6, the first eight relations are evidence relations. For example, we know that 'Ullman' and 'Stanford Univ.' co-occur in some web page, and that 'Doc201' is the homepage of 'Joe'. In addition to evidence, there are also relations whose content we do not know; they are called *query relations*. A user may ask for predictions on some or all query relations. For example, `affil` is a query relation since we want to predict affiliation relationships between persons and organizations. The other two query relations are `pCoref` and `oCoref`, for person and organization coreference, respectively.

In addition to schema and evidence, we also provide a set of MLN rules encoding our knowledge about the correlations and constraints over the relations. An MLN rule is a first-order logic formula associated with a real number (or infinity) called a *weight*. Infinite-weighted rules are called hard rules, which means that they must hold in any prediction that the MLN system makes. In contrast, rules with finite weights are soft rules: a positive weight indicates confidence in the rule's correctness.[3]

Example 2.2. *An important type of hard rule is a standard SQL query, e.g., to transform the results for use in the application. A more sophisticated example of hard rule is to encode that coreference has a transitive property, which is captured by the hard rule* F_3*. Rules* F_8 *and* F_9 *use person-organization co-occurrences (*`coOccurs`*) together with coreference (*`pCoref` *and* `oCoref`*) to deduce affiliation relationships (*`affil`*). These rules are soft since co-occurrence in a webpage does not necessarily imply affiliation.*

Intuitively, when a soft rule is violated, we pay a *cost* equal to the absolute value of its weight (described below). For example, if `coOccurs`('Ullman',

[3]Roughly these weights correspond to the log odds of the probability that the statement is true. (The log odds of probability p is $\log \frac{p}{1-p}$.) In general, these weights do not have a simple probabilistic interpretation [103].

'Stanford Univ.') and `pCoref`('Ullman', 'Jeff Ullman'), but not `affil`('Jeff Ullman', 'Stanford Univ.'), then we pay a cost of 4 because of F_9.

Similarly, affiliation relationships can be used to deduce non-obvious coreferences. For instance, using the fact that 'Mike' is affiliated with both 'UC-Berkeley' and 'UCB', ELEMENTARY may infer that 'UC-Berkeley' and 'UCB' refer to the same organization (rules on `oCoref` are omitted from Figure 2.6). If ELEMENTARY knows that 'Joe' co-occurs with 'UCB', then it is able to infer Joe's affiliation with 'UC-Berkeley'. Such interdependencies between predictions are sometimes called *joint inference*.

Semantics An MLN program defines a probability distribution over possible worlds. Formally, we first fix a schema σ (as in Figure 2.6) and a domain D. Given as input a set of formulae $\bar{F} = F_1, \ldots, F_N$ with weights $w_1, \ldots, w_N$, they define a probability distribution over *possible worlds* as follows. Given a formula F_k with free variables $\bar{x} = (x_1, \cdots, x_m)$, for each $\bar{d} \in D^m$ we create a new formula $g_{\bar{d}}$ called a *ground formula* where $g_{\bar{d}}$ denotes the result of substituting each variable x_i of F_k with d_i. We assign the weight w_k to $g_{\bar{d}}$. Denote by $G = (\bar{g}, w)$ the set of all such weighted ground formulae of $\bar{F}$. Essentially, G forms a Markov random field (MRF) [63] over a set of Boolean random variables (representing the truth value of each possible *ground tuple*). Let w be a function that maps each ground formula to its assigned weight. Fix an MLN $\bar{F}$, then for any possible world (instance) I, we say a ground formula g is *violated* if $w(g) > 0$ and g is false in I, or if $w(g) < 0$ and g is true in I. We denote the set of ground formulae violated in a world I as $V(I)$. The cost of the world I is

$$\text{cost}_{\text{MLN}}(I) = \sum_{g \in V(I)} |w(g)| \tag{2.1}$$

Through cost_{MLN}, an MLN defines a probability distribution over all instances using the exponential family of distributions (that are the basis for

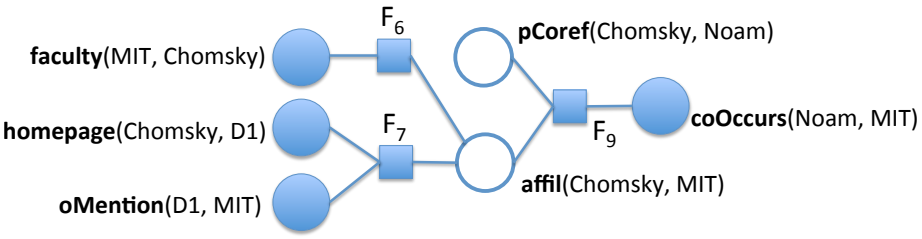


Figure 2.7: *A portion of the graphical model represented by the MLN in Figure 2.6.*

graphical models [137]):

$$\Pr[I] = Z^{-1} \exp\{-\text{cost}_{\text{MLN}}(I)\}$$

where Z is a normalizing constant.

Connection with Graphical Models An MLN essentially encodes a graphical model with X being all evidence tuples, and Y being all unknown tuples. Each MLN rule compactly represents a family of factors: let F_i be a formula with k literals, then F_i defines a factor function of the form $f_i : \{0,1\}^k \mapsto \{0, -|w_i|\}$. Each factor function takes value $-|w_i|$ if and only if the corresponding ground formulae is violated. Figure 2.7 illustrates a small portion of the graphical model represented by the MLN in Figure 2.6.

To better understand the semantics of MLNs, consider the following example.

Example 2.3. *If the* MLN *contains a single rule, then there is a simple probabilistic interpretation. Consider the rule* $R(`a')$ *with weight* $w$*. Then every possible world that contains* $R(`a')$ *has the same probability. Assuming that there are an equal number of possible worlds that contain* $R(`a')$ *and do not, then the probability that a world contains* $R(`a')$ *according to this distribution*

is $e^w/(e^w + 1)$. *In general, however, it is not possible to give such a direct probabilistic interpretation [103].*

2.3 Statistical Inference

There are two main tasks on a statistical model: (1) *inference*, namely the process of making predictions given evidence and a statistical model; and (2) *learning*, namely to find optimal correlation structure or parameters within a certain family of statistical models given example pairs of evidence and desired predictions. Because learning typically involves inference as subroutines, we present several common inference algorithms in this section and defer learning to the next section.

Inference Types There are two main types of inference with statistical models: *MAP (maximum a posterior) inference*, where we want to find a most likely assignment $Y = \mathbf{y}$, and *marginal inference*, where we want to compute the marginal probability of each unknown variable in Y. For some specialized statistical models such as LR and CRF, there are efficient algorithms for inference. For general graphical models or MLNs, both types of inference are intractable, but there are approximate algorithms.

2.3.1 Inference on Specialized Models

We describe inference algorithms for LR, CRF, and CC.

Logistic Regression Inference on LR models is straightforward [47]: one only needs to directly apply the model definition to compute the dot product between the weight vector $\mathbf{w}$ and evidence variables $\mathbf{x}$ and then calculate the logistic function $\frac{1}{1+e^{-x}}$. If the evidence data are stored in an RDBMS, one could also perform LR inference with user defined aggregates (UDA).

Algorithm 1 Viterbi algorithm for MAP inference on linear-chain CRFs [70]

Input: A linear-chain CRF model (using notations in Section 2.2.1)
Output: A most likely state $\mathbf{y}^* = \arg\max_{\mathbf{y}} \Pr[Y = \mathbf{y}]$

Initialize matrix $V(i, y) \leftarrow 0$ for $1 \leqslant i \leqslant n$ and $y \in \{0, 1\}$
Initialize base cases $V(1, y) \leftarrow \exp\{f(\mathbf{x}_1, 0, y)\}$ for $y \in \{0, 1\}$
for $k = 2$ to n **do**
  // Recurse according to the bigram factors
  $V(k, y) \leftarrow \max_{y'} V(k-1, y') \exp\{f(\mathbf{x}_k, y', y)\}$ for $y \in \{0, 1\}$
$\mathbf{y}^*_k \leftarrow \arg\max_y V(k, y)$ for $1 \leqslant k \leqslant n$

Algorithm 2 Forward-backward algorithm for marginal inference on linear-chain CRFs [70]

Input: A linear-chain CRF model (using notations in Section 2.2.1)
Output: Marginal probabilities $\Pr[Y_i = 1]$ for $1 \leqslant i \leqslant n$

Initialize forward matrix $A(i, y) \leftarrow 0$ for $1 \leqslant i \leqslant n$ and $y \in \{0, 1\}$
Initialize backward matrix $B(i, y) \leftarrow 0$ for $1 \leqslant i \leqslant n$ and $y \in \{0, 1\}$
Base cases $A(1, y) \leftarrow \exp\{f(\mathbf{x}_1, 0, y)\}$ for $y \in \{0, 1\}$
Base cases $B(n, y) \leftarrow \exp\{f(\mathbf{x}_n, 0, y)\}$ for $y \in \{0, 1\}$
// Forward recursion
for $k = 2$ to n **do**
  $A(k, y) = \sum_{y'} A(k-1, y') \exp\{f(\mathbf{x}_k, y', y)\}$ for $y \in \{0, 1\}$
// Backward recursion
for $k = n - 1$ to 1 **do**
  $B(k, y) = \sum_{y'} B(k+1, y') \exp\{f(\mathbf{x}_k, y', y)\}$ for $y \in \{0, 1\}$
// Calculate partitioning function
$Z \leftarrow \sum_y B(1, y)$
$\Pr[Y_i = 1] = A(i, 1)B(i, 1)/Z$ for $1 \leqslant i \leqslant n$

Conditional Random Fields Linear-chain CRFs can be efficiently solved with dynamic programming-based Viterbi algorithm (for MAP inference) and the forward-backward algorithm (for marginal inference) [70]. Following the notations in Section 2.2.1, we describe the algorithms in Algorithm 1 and Algorithm 2, respectively.

Algorithm 3 A simple algorithm for correlation clustering [2]

Input: A graph $G = (V, E)$ with weight function w (using notations in Section 2.2.2)
Output: A partitioning of $V = V_1, \ldots, V_k$ for some $1 \leqslant k \leqslant |V|$

Initialize node set $C = \emptyset$ // covered nodes
Initialize edge set $D = \emptyset$ // retained edges
Select a permutation π_V of V uniform at random
for each v in the order π_V **do**
  // Ignore nodes that have been covered
  if $v \notin C$ **then**
    // Group this node with all neighbors connected by a positive edge
    $D \leftarrow D \cup \{e = (v, u) \mid u \notin C, w(e) > 0\}$
    // Consider all nodes in this group as being covered
    $C \leftarrow C \cup \{v\} \cup \{u \mid e = (v, u) \in E, w(e) > 0\}$
return components in $H = (V, D)$

Correlation Clustering Coreference or correlation clustering is a well-studied problem [35], and there are many approximate inference techniques [2, 8, 119]. We list the pseudo-code of two such algorithms in Algorithm 3 and Algorithm 4.

2.3.2 Inference on Graphical Models

In general, inference on graphical models is intractable. But if a graphical model is a tree (in the factor graph representation), there are efficient dynamic programming algorithms that perform exact inference. We de-

Algorithm 4 An MCMC-based algorithm for correlation clustering [119]

Input: A graph $G = (V, E)$ with weight function w (see Section 2.2.2); $M \in \mathbb{N}$
Output: A partitioning of $V = V_1, \ldots, V_k$ for some $1 \leqslant k \leqslant |V|$
1: Initialize singleton clusters $\mathcal{C} = \{C_v = \{v\} \mid v \in V\}$
2: **for** $i = 1$ to M **do**
3: Randomly pick a non-empty cluster $S \in \mathcal{C}$, a node $v \in S$, and another cluster $T \neq S$
4: Calculate the change of $\text{cost}_{\text{coref}}$ (denoted δ) for moving v from S to T
5: With probability $\min(e^{-\delta}, 1)$, move v from S to T
6: // The move is greedy: cost-decreasing moves are always taken
7: **return** non-empty clusters in $\mathcal{C}$

scribe inference algorithms for both this special case and general graphical models.

Special Cases: Trees The inference algorithms for tree-shaped graphical models are essentially generalizations of the Viterbi and forward-backward algorithms for CRFs. They pass messages between the nodes in a graphical model. The MAP inference algorithm for tree-shaped graphical models is called MaxProduct, and the marginal inference algorithm is called SumProduct. We display the SumProduct algorithn in Algorithm 5. The MaxProduct algorithm is similar to SumProduct but replaces the sum operations with max and adds a backtracking step at the end of the algorithm.

General Graphical Models For graphical models that contain cycles, the above SumProduct and MaxProduct no longer apply. There are several common techniques to perform approximate inference on such graphs. First, one could apply the junction tree technique to transform the graph into a tree by contracting subsets of nodes in the graph. Second, a popular approach is called (loopy) belief propagation, where we run the SumProd-

Algorithm 5 SumProduct algorithm for marginal inference on graphical models [67]

Input: A tree-shaped factor graph $T = (V, F, E)$ where $V = X \cup Y$ (see Section 2.2.2)

Output: $\Pr[Y_i = 1]$ for $Y_i \in Y$

Designate an arbitrary tree root $Y_r \in Y$
Denote by $\mathrm{par}(N)$ (resp. $\mathrm{chd}(N)$) the parent (resp. children) of a node N
// Set up base cases
For each edge $e = (v, f) \in E$, for $y \in \{0, 1\}$, allocate $M(v, f, y) \in \mathbb{R}$ and $M(f, v, y) \in \mathbb{R}$
For each leaf variable node $Y_i \in Y$, $M(Y_i, \_, \_) \leftarrow 1$
For each leaf factor node $A = \{v\} \in F$, $M(A, v, y) = f_A(y)$ for $y \in \{0, 1\}$
// First pass: from leaves toward root
while $\exists N \in V$ s.t. all $M(\mathrm{chd}(N), N, \_)$ entries are filled except for $M(\mathrm{par}(N), N, \_)$ **do**
  If $N \in Y$ then $M(N, \mathrm{par}(N), y) \leftarrow \prod_{K:K \neq \mathrm{par}(N)} M(K, N, y)$ for $y \in \{0, 1\}$
  If $N \in F$ and $P \equiv \mathrm{par}(N) \in Y$, then for $y \in \{0, 1\}$ $M(N, P, y) \leftarrow \sum_{\mathbf{v}_{N \setminus \{P\}}} f_N(N \mid_{P=y}) \prod_{Q \in N \setminus \{P\}} M(Q, N, \mathbf{v}_Q)$ where $\mathbf{v} = \mathbf{x} \cup \mathbf{y}$
// Second pass: from root toward leaves
$M(Y_r, \_, \_) \leftarrow 1$
while $\exists N \in V$ s.t. $M(N, \mathrm{chd}(N), \_)$ entries are empty but $M(\mathrm{par}(N), N, \_)$ is filled **do**
  for each $P \in \mathrm{chd}(N)$ **do**
    If $N \in Y$ then $M(N, P, y) \leftarrow \prod_{K:K \neq P} M(K, N, y)$ for $y \in \{0, 1\}$
    If $N \in F$, then for $y \in \{0, 1\}$ $M(N, P, y) \leftarrow \sum_{\mathbf{v}_{N \setminus \{P\}}} f_N(N \mid_{P=y}) \prod_{Q \in N \setminus \{P\}} M(Q, N, \mathbf{v}_Q)$ where $\mathbf{v} = \mathbf{x} \cup \mathbf{y}$
// Now every node has received messages from all other nodes
$\Pr[Y_i = 1] \leftarrow \prod_A M(A, Y_i, 1) / \{\sum_{y \in \{0,1\}} \prod_A M(A, Y_i, 1)\}$

Algorithm 6 Gibbs sampling for marginal inference on graphical models [63]

Input: A factor graph $G = (V, F, E)$ where $V = X \cup Y$; $M \in \mathbb{N}$
Output: M samples of Y
1: Initialize Y with a random assignment $\mathbf{y}^0$
2: // One sample per iteration
3: **for** $i = 1$ to M **do**
4: $\quad \mathbf{y}^i \leftarrow \mathbf{y}^{i-1}$
5: $\quad$ // The order could be randomized
6: $\quad$ **for** each variable $Y_j \in Y$ **do**
7: $\quad\quad$ Sample $\mathbf{y}_j^i$ according to $\Pr[Y_j | \mathbf{y}^i_{-j}]$ where $\mathbf{y}^i_{-j} = \mathbf{y}^i \setminus \{\mathbf{y}_j\}$
8: **return** $\mathbf{y}_i$ for $1 \leqslant i \leqslant M$

uct protocol regardless of the fact that there are cycles in the graph. There exist many optimizations for belief propagation but in general there are no theoretical guarantees with belief propagation. A third approach is to employ sampling-based algorithms, e.g., Monte Carlo Markov Chain (MCMC) algorithms. As an example, we list the pseudo-code of a popular MCMC algorithm called Gibbs sampling in Algorithm 6. The conditional probability $\Pr[Y_j | \mathbf{y}^i_{-j}]$ can be efficiently computed because it depends on only the *Markov blanket* of Y_j, namely variables that share a factor with Y_j. Two filtering techniques are sometimes applied to Gibbs sampling to improve the sampling quality: *burn-in*, where we discard the first few samples, and *thinning*, where we retain every k samples for some integer k.

2.3.3 Inference on Markov Logic

We describe state-of-the-art approaches to MLN inference as implemented in Alchemy.

Algorithm 7 LazyGrounding [100]

Input: an MLN and atom a to be activated
Input: A_c: currently active ground atoms
Input: G_c: currently active ground clauses
Output: expanded A_c and G_c

1: $G \leftarrow$ ground clauses that can be violated by some assignment to a and currently active atoms
2: $A_c \leftarrow A_c \cup \{a\}$
3: $G_c \leftarrow G_c \cup G$

Algorithm 8 WalkSAT [58, 100]

Input: A: initial active ground atoms
Input: C: initial active ground clauses
Input: `MaxFlips`, `MaxTries`
Output: σ^*: a truth assignment to A

1: `lowCost` $\leftarrow +\infty$, $\sigma^* \leftarrow \mathbf{0}$
2: **for** `try` $= 1$ to `MaxTries` **do**
3: $\sigma \leftarrow$ a random truth assignment to A
4: **for** `flip` $= 1$ to `MaxFlips` **do**
5: pick a random $c \in C$ that's violated
6: `rand` $\leftarrow$ random real $\in [0, 1]$
7: **if** `rand` $\leqslant 0.5$ **then**
8: `atom` $\leftarrow$ random atom $\in c$
9: **else**
10: `atom` $\leftarrow \arg\min_{a \in c} \delta(a)$
11: // $\delta(a) = \text{cost}_{\text{MLN}}(\sigma') - \text{cost}_{\text{MLN}}(\sigma)$
12: // where σ' is σ with atom a flipped
13: **if** `atom` is inactive **then**
14: activate `atom`; expand A, C
15: // Keep track of best world seen
16: flip `atom` in σ
17: **if** $\text{cost}_{\text{MLN}}(\sigma) <$ `lowCost` **then**
18: `lowCost` $\leftarrow \text{cost}_{\text{MLN}}(\sigma)$, $\sigma^* \leftarrow \sigma$
19: **return** σ^*

Lazy Grounding Conceptually, we might ground an MLN formula by enumerating all possible assignments to its free variables. However, this is both impractical and unnecessary. Fortunately, in practice a vast majority of ground clauses are satisfied by evidence regardless of the assignments to unknown truth values; we can safely discard such clauses [115]. Pushing this idea further, Poon et al. [100] proposed a method called "lazy inference" which is implemented in Alchemy. Specifically, Alchemy works under the more aggressive hypothesis that most atoms will be false in the final solution, and in fact throughout the entire execution. To make this idea precise, call a ground clause *active* if it can be violated by flipping zero or more active atoms, where an atom is active if its value *flips* at any point during execution (see Algorithm 7). Alchemy keeps only *active ground clauses* in memory, which can be much smaller than the full set of ground clauses. Furthermore, as on-the-fly incremental grounding is more expensive than batch grounding, Alchemy uses the following one-step look-ahead strategy at the beginning: assume all atoms are inactive and compute active clauses; *activate* the atoms that appear in any of the active clauses; and recompute active clauses.This "look-ahead" procedure could be repeatedly applied until convergence, resulting in an *active closure*. Tuffy implements this closure algorithm.

MAP Inference with WalkSAT Because the factors of the graphical model generated from an MLN are all conjunctive normal forms (CNF) or disjunctive normal forms (DNF), MAP inference in MLNs can be reduced to the classic MaxSAT problem. The state-of-the-art MAP inference algorithm for MLNs is a heuristic search algorithm called WalkSAT [58] (see Algorithm 8). WalkSAT performs random walks over all possible assignments to Y (i.e., unknown atoms in the MLN). At each step, it randomly picks a ground clause that is violated and fixes it in one of two ways: flipping a randomly selected variable in this clause, or flipping the

variable in this clause that would result in the largest drop in the total cost.

Algorithm 9 SampleSAT [139]

Input: atoms A and clauses M
Input: N // number of steps
Output: σ^*: a truth assignment to A

```
lowCost ← +∞, σ* ← 0
σ ← a random truth assignment to A
for i = 1 to N do
  pick a random c ∈ M that's violated
  rand ← random real ∈ [0, 1]
  if rand ⩽ 0.5 or cost_MLN(σ) > 0 then
    run a WalkSAT step (Algorithm 8)
  else
    randomly select an atom a in c
    δ ← change of cost if a is flipped.
    with probability e^{-δ}, flip a
    // Keep track of best world seen
    if cost_MLN(σ) < lowCost then
      lowCost ← cost_MLN(σ), σ* ← σ
return σ*
```

Marginal Inference with MC-SAT The state-of-the-art marginal inference algorithm for MLNs is MC-SAT [96]. Motivated by the fact that the factors compiled from an MLN are logical formulae, MC-SAT combines an MCMC style sampling algorithm with a heuristic SAT sampling algorithm called SampleSAT [139]. The SampleSAT algorithm in turn combines WalkSAT with simulated annealing. The pseudo-code for SampleSAT and MC-SAT is listed in Algorithm 9 and Algorithm 10, respectively.

2.3.4 Dual Decomposition

A classic inference technique is *dual decomposition* [66, 123] that can decompose a general intractable statistical model (e.g., a graphical model) into

Algorithm 10 MC-SAT [96]

Input: atoms A and clauses C
Input: N // number of samples
Output: N samples of assignments to A

$\sigma_0 \leftarrow$ a random assignment to A satisfying all hard clauses in C
for $i = 1$ to N **do**
  $M \leftarrow \emptyset$
  // Sample from current "good"
  // clauses for next iteration
  for all $c \in C$ not violated by σ_{i-1} **do**
    with probability $1 - e^{-|w(c)|}$,
    $M \leftarrow M \cup \{c\}$
  $\sigma_i \leftarrow$ running SampleSAT on M
return σ_i for $1 \leqslant i \leqslant N$

multiple pieces such that each piece (e.g., a tree) may become tractable. We illustrate the basic idea of dual decomposition with a simple example. Consider the problem of minimizing a real-valued function $f(x_1, x_2, x_3)$. Suppose that f can be written as

$$f(x_1, x_2, x_3) = f_1(x_1, x_2) + f_2(x_2, x_3).$$

Further suppose that we have black boxes to solve f_1 and f_2 (plus linear terms). To apply these black boxes to minimize f we need to cope with the fact that f_1 and f_2 share the variable x_2. Following dual decomposition, we can rewrite $\min_{x_1,x_2,x_3} f(x_1, x_2, x_3)$ into the form

$$\min_{x_1,x_{21},x_{22},x_3} f_1(x_1, x_{21}) + f_2(x_{22}, x_3) \text{ s.t. } x_{21} = x_{22},$$

where we essentially made two copies of x_2 and enforce that they are identical. The significance of such rewriting is that we can apply Lagrangian relaxation to the equality constraint to decompose the formula into two independent pieces. To do this, we introduce a scalar variable $\lambda \in \mathbb{R}$

(called a *Lagrange multiplier*) and define $g(\lambda) =$

$$\min_{x_1,x_{21},x_{22},x_3} f_1(x_1, x_{21}) + f_2(x_{22}, x_3) + \lambda(x_{21} - x_{22}).$$

For any λ, we have $g(\lambda) \leqslant \min_{x_1,x_2,x_3} f(x_1, x_2, x_3)$.[4] Thus, the tightest bound is to maximize $g(\lambda)$; the problem $\max_\lambda g(\lambda)$ is a *dual problem* for the problem on f. If the optimal solution of this dual problem is feasible (here, $x_{21} = x_{22}$), then the dual optimal solution is also an optimum of the original program [143, p. 168].

The key benefit of this relaxation is that, instead of a single problem on f, we can now compute $g(\lambda)$ by solving two independent problems (each problem is grouped by parentheses) that may be easier to solve:

$$\begin{aligned} g(\lambda) = &\min_{x_1,x_{21}} \left(f_1(x_1, x_{21}) + \lambda x_{21}\right) \\ &+ \min_{x_{22},x_3} \left(f_2(x_{22}, x_3) - \lambda x_{22}\right). \end{aligned}$$

To compute $\max_\lambda g(\lambda)$, we can use standard techniques such as *projected subgradient* [143, p. 174]. Notice that dual decomposition can be used for MLN inference if x_i are truth values of ground tuples and one defines f to be $\text{cost}_{\text{MLN}}(I)$ as in Equation 2.1.

2.4 Statistical Learning

The statistical models in Section 2.2 have parameters such as the weight vector in logistic regression and conditional random fields and the rule weights in an MLN. To determine the "optimal" value for these parameters, a common approach is to apply statistical learning. In addition to the model $\mathcal{M}_w$ itself (with parameters w to be determined), the input also

[4] One can always take $x_{21} = x_{22} = x_2$ in the minimization, and the value of the two objective functions are equal.

includes a set of *training examples* $T = \{(x_i, y_i)\}_{i=1}^{m}$ and an *objective function* $h_T(w) : \mathbb{R}^{|w|} \mapsto \mathbb{R}$. The goal is to find $\arg\min_w h_T(w)$.

To illustrate, consider a logistic regression model with the following objective function

$$h_T(w) = \sum_{i=1}^{m} \log(1 + \exp(-y_i w^T x_i)) + \mu \|w\|_1.$$

Intuitively, the sum penalizes the "discrepancy" between the model's prediction (i.e., $w^T x_i$) and the ground-truth value in the example (i.e., y_i), while the second term $\mu\|w\|_1$ penalizes the "magnitude" of the parameters w. μ is a "meta-parameter" that intuitively controls how much restriction we place on the magnitude of the parameters (and actually also the "sparsity" of the parameters, i.e., how many entries of w are non-zero).

There are many algorithms for statistical learning. However, as this dissertation does not directly address learning algorithms themselves, we refer the reader to papers that do so; e.g., scaling statistical learning using an RDBMS [37], parallelization [38], weight learning for Markov logic [76], and structure learning for Markov logic [61, 62].

2.5 Additional Related Work

We discussed some background material and pioneering work on KBC in Section 2.1. In this section, we discuss additional related work.

Knowledge-base Construction using Statistical Inference There is a trend to build KBC systems with increasingly sophisticated statistical inference. For example, CMU's NELL [17] integrates four different extraction components that implement complementary techniques and consume various data resources (e.g., natural-language text, lists and tables in webpages, and human feedback). MPI's SOFIE/Prospera [84, 126] combines pattern-

based relation extraction approaches with domain-knowledge rules, and performs consistency checking against the existing YAGO knowledge base. Microsoft's StatsSnowball/EntityCube [149] performs iterative distant supervision while using the ℓ_1-norm regularization technique to reduce noisy extraction patterns; similar to Elementary, StatsSnowball also incorporates domain knowledge with Markov logic and observed quality benefit in doing so. Finally, behind IBM's DeepQA project's remarkable success at the *Jeopardy* Challenge, there is a "massively parallel probabilistic evidence-based architecture" that combines "more than 100 different techniques" [39].

Distant Supervision The idea of using entity-level structured data (e.g., facts in a database) to generate mention-level training data (e.g., in English text) is a classic one: researchers have used variants of this idea to extract entities of a certain type from webpages [15, 46]. More closely related to relation extraction is the work of Lin and Patel [74] that uses dependency paths to find answers that express the same relation as in a question.

Since Mintz et al. [82] coined the name *"distant supervision,"* there has been growing interest in this technique. For example, distant supervision has been used for the TAC-KBP slot-filling tasks [128] and other relation-extraction tasks [17, 51, 85, 86]. In contrast, we study how increasing input size (and incorporating human feedback) improves the result quality of distant supervision.

In Chapter 4, we focus on logistic regression, but it is interesting future work to study more sophisticated probabilistic models; such models have recently been used to relax various assumptions of distant supervision [50, 106, 147]. Specifically, they address the noisy assumption that, if two entities participate in a relation in a knowledge base, then all co-occurrences of these entities express this relation. In contrast, we explore the effectiveness of increasing the training data sizes to improve distant-

supervision quality.

Sheng et al. [117] and Gormley et al. [43] study the quality-control issue for collecting training labels via crowdsourcing. Their focus is the collection process; in contrast, our goal is to quantify the impact of this additional data source on distant-supervision quality. Moreover, we experiment with one order of magnitude more human labels. Hoffmann et al. [49] study how to acquire end-user feedback on relation-extraction results posted on an augmented Wikipedia site; it is interesting future work to integrate this source in our experiments. *Active learning* [113] is the idea of adaptively selecting *unlabeled* examples and ask for human labels. Intuitively one would expect the strategies in active learning to help with incorporating human feedback for distant supervision. In our preliminary experiments on distant supervision, we experimented with several state-of-the-art active learning heuristics [72, 114]. It is interesting to further explore the intersection between active learning and crowdsourcing.

Markov Logic There are statistical-logical frameworks similar to MLNs, such as Probabilistic Relational Models [41] and Relational Markov Models [131]. Inference on those models also requires grounding and search, and we are optimistic that the lessons we learned with MLNs carry over to these models. MLNs are an integral part of state-of-the-art approaches in a variety of applications: natural language processing [105], ontology matching [145], information extraction [97], entity resolution [120], etc. And so, there is an application push to support MLNs.

Markov Logic Inference The idea of using the stochastic local search algorithm WalkSAT to find the most likely world is due to Kautz et al. [58]. Singla and Domingos [121] proposed lazy grounding and applies it to WalkSAT, resulting in an algorithm called LazySAT that is implemented in Alchemy. The idea of ignoring ground clauses that are satisfied by evidence is highlighted as an effective way of speeding up the MLN

grounding process in Shavlik and Natarajan [115], which formulates the grounding process as nested loops and provides heuristics to approximate the optimal looping order. Mihalkova and Mooney [80] also employ a bottom-up approach as does Tuffy, but they address structure learning of MLNs whereas we focus on inference. Knowledge-based model construction [142] is a technique that, given a query, finds the minimal relevant portion of a graph; although the resulting subgraph may contain multiple components, the downstream inference algorithm may not be aware of it and thereby cannot benefit from the speedup in Theorem 5.2.

While Tuffy employs the simple WalkSAT algorithm, there are more advanced techniques for MAP inference [32, 45]. For hypergraph partitioning, there are established solutions such as hMETIS [56]. However, existing implementations of them are limited by memory size, and it is future work to adapt these algorithms to on-disk data; this motivated us to design the partitioning algorithm in Tuffy. The technique of *cutset conditioning* [94] from the SAT and probabilistic inference literature is closely related to our partitioning technique [3, 93]. Cutset conditioning recursively conditions on cutsets of graphical models, and at each step exhaustively enumerates all configurations of the cut, which is impractical in our scenario: even for small datasets, the cut size can easily be thousands, making exhaustive enumeration infeasible. Instead, we use a Gauss-Seidel strategy, which we show is efficient and effective. Additionally, our conceptual goals are different: our goal is to find an analytic formula that quantifies the effect of partitioning and then, we use this formula to optimize the IO and scheduling behavior of a class of local search algorithms; in contrast, prior work focuses on designing new inference algorithms.

Scaling Markov Logic There has been extensive research interest in techniques to improve the scalability and performance of MLN inference. For example, multiple researchers have studied how to prune MRFs based on

logical rules in MLNs [115, 121], and incrementally ground or unroll the MRF [104, 122]. Mihalkova and Richardson [81] study how to avoid redundant computation by clustering similar query literals. It is an interesting problem to incorporate their techniques into Tuffy.

An orthogonal direction to dual decomposition is *lifted inference* [14, 25, 59, 95, 111]. Lifted inference is similar to our proposed approach for dual decomposition in that both exploit symmetries in the MRF. Nevertheless, while Felix uses symmetries to invoke specialized inference algorithms, lifted inference augments a general inference algorithm to operate on first-order (non-grounded) representations. For example, Singla and Domingos [122] proposed a lifted version of belief propagation to exploit these symmetries [81, 122].

Probabilistic Databases Pushing statistical reasoning models inside a database system has been a goal of many projects [29, 52, 55, 102, 138]. Most closely related is the BayesStore project, in which the database essentially stores *Bayes nets* [94] and allows these networks to be retrieved for inference by an external program. In contrast, Tuffy uses an RDBMS to optimize the inference procedure. The Monte-Carlo database [52] made sampling a first-class citizen inside an RDBMS. In contrast, in Tuffy our approach can be viewed as pushing classical search inside the database engine. One way to view an MLN is a compact specification of *factor graphs* [112]. Sen et al. [112] proposed new algorithms; in contrast, we take an existing, widely used class of algorithms (local search), and our focus is to leverage the RDBMS to improve performance. There has also been an extensive amount of work on *probabilistic databases* [6, 10, 24, 101] that deal with simpler probabilistic models. Finding the most likely world is trivial in these models; in contrast, it is highly non-trivial in MLNs (in fact, it is NP-hard [30]). Finally, none of these prior approaches deal with the core technical challenge Tuffy addresses, which is handling AI-style search

inside a database.

Dual Decomposition Dual decomposition is a classic and general technique in optimization that decomposes an objective function into multiple smaller subproblems; in turn these subproblems communicate to optimize a global objective via Lagrange multipliers [11]. Recently dual decomposition has been applied to inference in graphical models such as MRFs. In the master-slave scheme [65, 66], the MAP solution from each subproblem is communicated and the Lagrange multipliers are updated with the projected gradient method at each iteration. Our prototype implementation of Felix uses the master-slave scheme. It is future work to adapt the closely related tree-reweighted (TRW) algorithms [64, 136] that decompose an MRF into a convex combination of spanning trees each of which can be solved efficiently.

Researchers have also applied dual decomposition in settings besides graphical models. For example, Rush et al. [108] employ dual decomposition to jointly perform tagging and parsing in natural language processing. Felix can be viewed as extending this line of work to higher-level tasks (e.g., classification and clustering). Dual decomposition requires iterative message passing to resolve the predictions output by subproblems. There is recent work on techniques to improve the convergence rate of dual decomposition [42, 54].

3 Knowledge-base Construction in Elementary

To perform web-scale KBC that combines diverse data resources and techniques, it is crucial to have a clean conceptual model for KBC and an infrastructure that is able to deal with terabytes of data. We first describe a simple KBC model that we use in Elementary, and then briefly discuss the infrastructure that enables web-scale KBC in Elementary. The work in this chapter appears in Niu, Zhang, Ré, and Shavlik [90].

3.1 Conceptual Model and Architecture

Conceptual Model Elementary adopts the classic Entity-Relationship (ER) model [16, 20]: the schema of the targe knowledge base (KB) is specified by an ER graph $G = (\bar{E}, \bar{R})$ where $\bar{E}$ is one or more sets of *entities* (e.g., people and organizations), and $\bar{R}$ is a set of relationships. Define $\mathcal{E}(G) = \cup_{E \in \bar{E}} E$, i.e., the set of known entities. To specify a KBC task to Elementary, one provides the schema G and a corpus D. Each document $d_i \in D$ consists of a set of (possibly overlapping) *text spans* (e.g., tokens or sentences) $T(d_i)$. Text spans referring to entities or relationships are called *mentions* (see Figure 3.1). Define $\mathcal{T}(D) = \cup_{d_i \in D} T(d_i)$. Our goal is to accurately populate the following tables:

- Entity-mention table $M(\mathcal{E}(G), \mathcal{T}(D))$.[1]
- Relationship-mention tables $M_{R_i} \subseteq \mathcal{T}(D)^{k+1}$ for each $R_i \in \bar{R}$; k is the arity of R_i, the first k attributes are entity mentions, and the last attribute is a relationship mention.

[1] For simplicity, in this conceptual model we assume that all entities are known, but Elementary supports generating novel entities for the KB as well (e.g., by clustering "dangling" mentions).

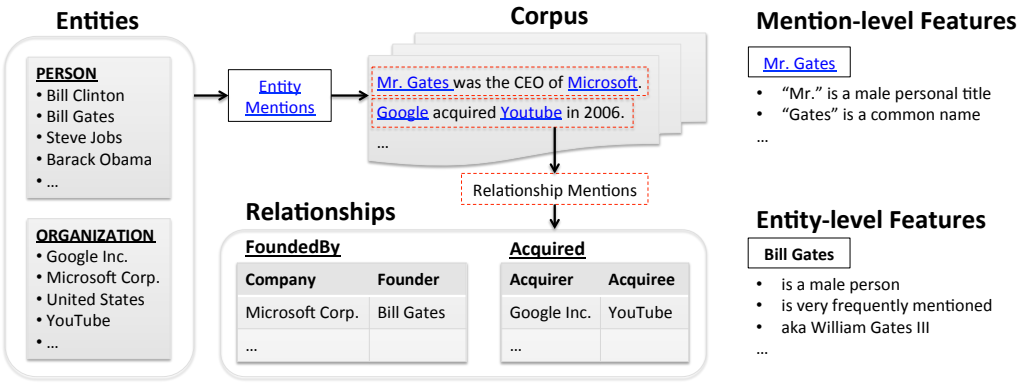


Figure 3.1: *An illustration of the KBC model in* ELEMENTARY.

- Relationship tables $R_i \in \bar{R}$.

Note that R_i can be derived from M_E and M_{R_i}. By the same token, M_E and M_{R_i} provide provenance that connects the KB back to the documents supporting each fact. The process of populating M_E is called *entity linking*; the process of populating M_{R_i} is called *relation extraction*. Intuitively, the goal is to produce an instance J of these tables that is as large as possible (high recall) and as correct as possible (high precision). ELEMENTARY populates the target KB based on signals from mention-level features (over text spans) and entity-level features (over the target KB). Mention-level features include, e.g., relative position in a document, matched regular expressions, web search results, etc. Entity-level features include, e.g., canonical names and known entity attributes (e.g., age, gender, alias, etc.).

Depending on the data sources and the feature-extraction process, there may be errors or inconsistencies among the feature values. These features are the input to ELEMENTARY's machine learning and statistical inference components. Thus, it is crucial that ELEMENTARY performs feature extraction in a scalable and efficient manner. A key issue for machine learning is the availability of training data, and an increasingly popular technique addressing this issue is distant supervision. We found that scal-

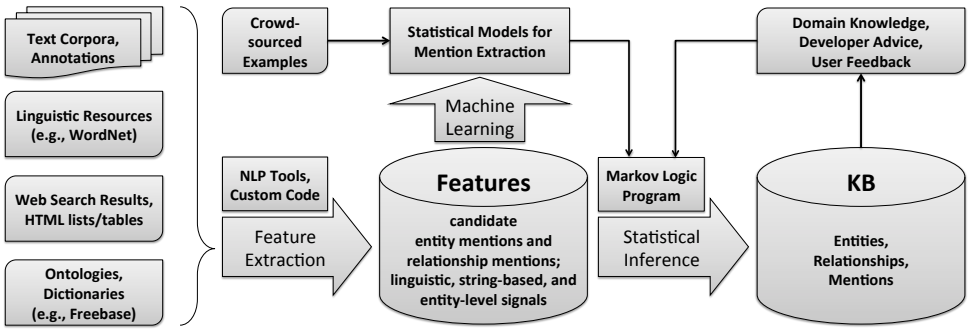


Figure 3.2: *Architecture of* Elementary. Elementary *takes as input diverse data resources, converts them into relational features, and then performs machine learning and statistical inference to construct knowledge bases.*

ing the input data resources is indeed an effective approach to improving the quality of KBC via distant supervision [148].

Elementary Architecture The central statistical modeling language in Elementary, Markov logic, operates on relational data. To accommodate diverse information sources, Elementary employs a two-phase architecture when processing a KBC task: feature extraction and statistical inference (see Figure 3.2). Intuitively, feature extraction concerns what signals may contribute to KBC and how to generate them from input data resources (e.g., pattern matching and selection), while statistical inference concerns what (deterministic or statistical) correlations and constraints over the signals are valuable to KBC and how to efficiently resolve them. The model for statistical inference, i.e., a Markov logic program, can be constructed using a combination of machine learning and domain knowledge.

All input data resources are first transformed into relational data via the feature-extraction step. For example, one may employ standard NLP tools to decode the structure (e.g., part-of-speech tags and parse trees) in text, run pattern matching to identify candidate entity mentions, perform topic modeling to provide additional features for documents or text spans,

etc. If a KBC technique (e.g., a particular learning or inference algorithm) can be expressed in MLNs, one can also translate them into MLN rules. Otherwise, one can execute the technique in the feature-extraction step and "materialize" its result in evidence or MLN rules. Once we have converted all signals into relational evidence, the second step is to perform statistical inference to construct a knowledge base as described in the above KBC model. From our experience of building large-scale knowledge bases and applications on top of them, we have found that it is critical to efficiently process structured queries over large volumes of structured data. Therefore, we have built Elementary on top of a relational database management system (RDBMS).

Development Workflow The high-level steps in applying Elementary to a KBC task are as follows:

1. Choose the entities and relationships to recognize in text.[2]
2. Choose a corpus of documents, and collect all data resources that may help with the current KBC task.
3. Perform feature extraction on the input data by running standard NLP tools or custom code to create a relational representation of all input data, i.e., evidence. Perform machine learning on the evidence if needed.
4. Create an MLN program that integrates the statistical models generated by machine learning, domain-knowledge rules, or other sources of information on the evidence and the target KB. If needed, perform weight learning.
5. Run statistical inference with the MLN program and the evidence to make predictions for the target KB.

[2]We leave open information extraction [146] as future work.

6. Inspect results and if appropriate, go to (3) or (4).

3.2 Examples

To demonstrate the flexibility of ELEMENTARY for KBC, we use several examples to illustrate how ELEMENTARY integrates diverse data resources and different techniques for KBC. Recall that there are two phases in ELEMENTARY: feature extraction and statistical inference. When integrating an existing KBC technique, one needs to decide what goes into feature extraction and what goes into statistical inference. To illustrate, we consider several common techniques for entity linking, relation extraction, and incorporating domain knowledge, respectively.

Entity Linking Entity linking is the task of mapping text spans to entities, i.e., populating the $M_E(E, \mathcal{T}(D))$ tables in the formal model. To identify candidate mentions of entities, there are several common techniques, e.g., (1) perform string matching against dictionaries; (2) use regular expressions or trigger words (e.g., "Prof." for person mentions); (3) run named-entity-recognition tools. All these techniques can be implemented in the feature-extraction phase of ELEMENTARY. Sometimes a mention may correspond to multiple candidate entities. To determine which entity is correct, one could use various techniques and data resources. For example, a heuristic is that "if the string of a mention is identical to the canonical name of an entity, then this mention is likely to refer to this entity"; one can express this heuristic in a (soft or hard) MLN rule:

$$\begin{aligned} &\text{MentionText}(mention, string) \\ \wedge\ &\text{EntityName}(entity, string) \\ \Rightarrow\ &M_E(entity, mention). \end{aligned}$$

When a named-entity-recognition (NER) tool is used in feature extraction, one can use the mention types output by the tool as a constraint for entity linking. For example, let NerMentionType($mention$, $type$) be an evidence relation storing NER output; then we can add the constraint

$$\begin{aligned} & \mathrm{M}_{\texttt{PERSON}}(entity, mention) \\ => & \text{ NerMentionType}(mention, \texttt{PERSON}). \end{aligned}$$

As another example, let Anchor($string$, $entity$, $freq$) be an evidence relation indicating how frequently (measured by the numeric attribute $freq$) an anchor text $string$ links to the Wikipedia page representing entity $entity$. Then one could use these signals for entity disambiguity with the rule

$$\begin{aligned} wgt(freq) : & \text{ Anchor}(string, entity, freq) \\ & \wedge \text{ MentionText}(mention, string) \\ => & \mathrm{M}_{\mathrm{E}}(entity, mention). \end{aligned}$$

where the $wgt(freq)$ syntax means that the rule weight is a function of the $freq$ attribute of Anchor. Note that one can also adapt Anchor for other data resources, e.g., web search results with mention phrases as queries and Wikipedia links as proxies for entities. Another common technique for entity linking is coreference. For example, let SamePerson($m1$, $m2$) be an equivalence relation indicating which pairs of person mentions are coreferent, i.e., referring to the same entity. Then one can use the following heuristics to propagate entity linking results between coreferent

mentions[3]:

$$\begin{aligned}
&\text{MentionText}(m1, s) \wedge \text{MentionText}(m2, s) \\
\wedge\ &\text{InSameDoc}(m1, m2) \wedge \text{FullName}(s) \\
\Rightarrow\ &\text{SamePerson}(m1, m2). \\
&\text{MentionText}(m1, s1) \wedge \text{MentionText}(m2, s2) \\
\wedge\ &\text{InSamePara}(m1, m2) \wedge \text{ShortFor}(s1, s2) \\
\Rightarrow\ &\text{SamePerson}(m1, m2). \\
&M_{\texttt{PERSON}}(e, m1) \wedge \text{SamePerson}(m1, m2) \\
\Rightarrow\ &M_{\texttt{PERSON}}(e, m2).
\end{aligned}$$

Relation Extraction Relation extraction is the task of determining whether a relationship is mentioned in a text span, i.e., populating the relationship mention tables M_{R_i}. For natural-language text, the most common approach is to perform classification based on linguistic patterns. Researchers have found many different kinds of linguistic patterns to be helpful; e.g., shallow features such as word sequences and part-of-speech tags between entity mentions [17, 149], deep linguistic features such as dependency paths [73, 82, 146], and other features such as the n-gram-itemset [84]. To improve coverage and reduce noisiness of these patterns, researchers have also invented different feature selection and expansion techniques; e.g., frequency-based filtering [84, 146], feature selection with ℓ_1-norm regularization [149], and feature expansion based on pattern similarity [73, 84]. In ELEMENTARY, one can implement different combinations of the above techniques in the feature extraction phase, and incorporate

[3]The weights are not shown for clarity.

the signals with (soft or hard) rules of the form

$$\begin{aligned} &\text{WordSequence}(s, m1, m2, \text{“was born in”}) \\ \Rightarrow\ &M_{\texttt{BirthPlace}}(m1, m2, s). \end{aligned}$$

where s is a sentence and $m1$ and $m2$ are two entity mentions (i.e., text spans) within s.

To learn the association between linguistic patterns and relationships, there are two common approaches: *direct supervision* and *distant supervision* [82]. In direct supervision, one uses mention-level annotations of relationship mentions as training data to learn a statistical model between patterns and relationships [70]. As mention-level annotations are usually rare and expensive to obtain, an increasingly popular approach is distant supervision, where one uses entity-level relationships and entity linking to heuristically collect silver-standard annotations from a text corpus [17, 82, 84, 149]. To illustrate, let KnownBirthPlace(person, location) be an existing knowledge base containing tuples like

$$\text{KnownBirthPlace}(\texttt{Barack\_Obama}, \texttt{Hawaii}),$$

then one can perform distant supervision in Elementary by learning weights for MLN rules of the form

$$\begin{aligned} wgt(pat) : &\text{WordSequence}(s, m1, m2, pat) \\ &\wedge M_{\text{PERSON}}(per, m1) \\ &\wedge M_{\text{LOCATION}}(loc, m2) \\ \Rightarrow\ &\text{KnownBirthPlace}(per, loc). \end{aligned}$$

where $w(pat)$ is a parameterized weight that depends on the pattern pat; intuitively it models how indicative this pattern is for the BirthPlace relationship. For example, Elementary may learn that “was born in” is indica-

tive of BirthPlace based on the tuple KnownBirthPlace(Barack_Obama, Hawaii), the sentence s ="Obama was born in Hawaii," and entity linking results such as $M_{\texttt{PERSON}}(\texttt{Barack\_Obama}, \texttt{Obama})$.

Domain Knowledge Several recent projects [17, 84, 149] show that entity-level resources can effectively improve the precision of KBC. For example, Prospera [84] found that validating the argument types of relationships can improve the quality of relation extraction. One can incorporate such constraints with rules like

$$\infty : M_{\texttt{BirthPlace}}(m1, m2, s) => \exists\, e1 \;\; M_{\texttt{PERSON}}(e1, m1)$$

which enforces that the first argument of a BirthPlace mention must be a person entity. Besides typing, one can also specify semantic constraints on relationships such as "a person can have at most one birth place" and "HasChild and HasParent are symmetric":

$$\begin{aligned} \infty :&\;\; \text{BirthPlace}(p, l1) \wedge l1 \neq l2 => \neg\text{BirthPlace}(p, l2). \\ \infty :&\;\; \text{HasChild}(p1, p2) <=> \text{HasParent}(p2, p1). \end{aligned}$$

Another type of domain knowledge involves corrections to systematic errors in the statistical extraction models. For example, from sentences like "Obama lived in Hawaii for seven years," distant supervision may erroneously conclude that the linguistic pattern "lived in" is strongly indicative of BirthPlace. Once a developer identifies such an error (say via debugging), one could correct this error by simply adding the rule

$$\begin{aligned} &\infty : \text{WordSequence}(s, m1, m2, \text{“}\texttt{lived in}\text{”}) \\ &=> \neg M_{\texttt{BirthPlace}}(m1, m2, s). \end{aligned}$$

Note that this rule only excludes mention-level relationships. Thus,

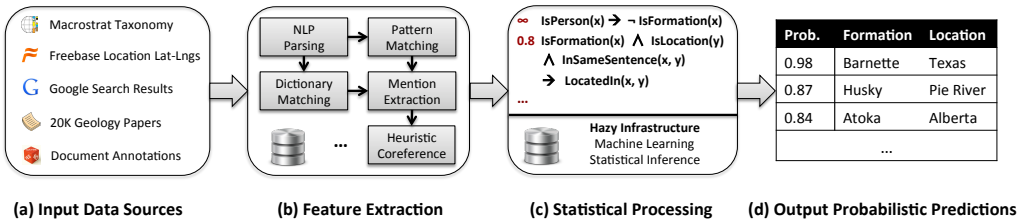


Figure 3.3: *An overview of* GeoDeepDive*'s development pipeline.*

if there are sentences "X lived in Y" and "X was born in Y", the final (entity-level) KB may still contain the tuple $\texttt{BirthPlace}(e_X, e_Y)$ based on the second sentence, where e_X (resp. e_Y) denotes the entity referred to by X (resp. Y).

3.2.1 Case Study: GeoDeepDive

To further illustrate Elementary's development process, let us walk through GeoDeepDive, a demo project where we collaborate with geoscientists to perform deep linguistic and statistical analysis over a corpus of tens of thousands of research papers in geology. The goal is to extract useful information from this corpus and organize it in a way that facilitates geologists' research. The current version of GeoDeepDive extracts mentions of rock formations, tries to assign various types of attributes to these formation mentions (e.g., location, time interval, carbon measurements), and then organizes the extractions and documents in spatial and temporal dimensions for geology researchers. Figure 3.3 shows a high-level overview of how we built GeoDeepDive with Elementary:

Input Data Sources Elementary embraces all data sources that can be useful for an application. For GeoDeepDive, we use the Macrostrat taxonomy[4] because it provides the set of entities of interest as well as domain-

[4] `http://macrostrat.org/`

specific constraints (e.g., a formation can only be associated with certain time intervals). We use Google search results to map location mentions to their canonical names and then to latitude-longitude (lat-lng) coordinates using Freebase[5]. These coordinates can be used to perform geographical matching against the formations' canonical locations (lat-lng polygons in Macrostrat). There are also (manual) document annotations of textual mentions of formation measurements that serve as training data.

Feature Extraction The input data sources may not have the desired format or semantics to be directly used as *signals* (or *features*) for statistical inference or learning. The feature extraction step performs such conversions. The developer explicitly specifies the schema of all relations, provides individual extractors, and then specifies how these extractors are composed together. For example, we (the developers) perform NLP parsing on the input corpus to produce per-sentence structured data such as part-of-speech tags and dependency paths. We then use the Macrostrat taxonomy and heuristics to extract candidate entity mentions (of formations, measures, etc.) as well as possible coreference relationships between the mentions.

Statistical Processing The signals produced by feature extraction may contain imprecision or inconsistency. To make coherent predictions, the developer provides constraints and (probabilistic) correlations over the signals. For example, we use the Markov logic language; a Markov logic program consists of a set of weighted logical rules and these rules represent high-level constraints or correlations.

Output Probabilistic Predictions The output from Hazy's statistical processing infrastructure is probabilistic predictions on relations of interest

[5] `http://freebase.com`

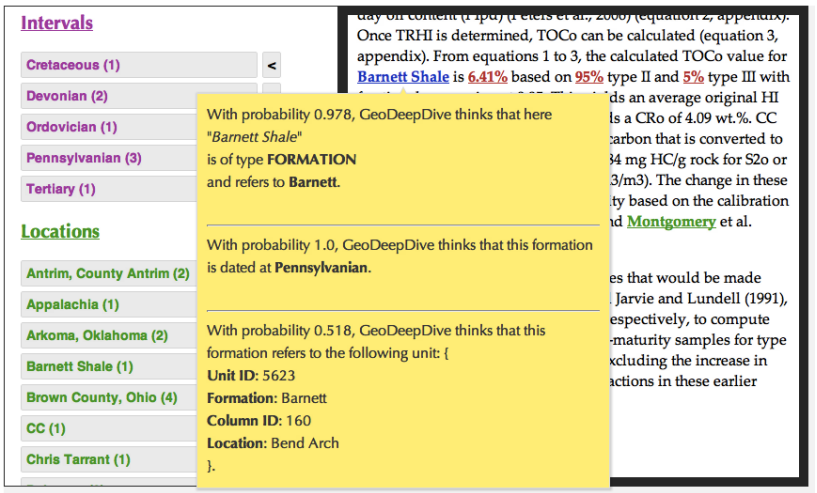


Figure 3.4: *Screen-shot from* GeoDeepDive

(e.g., `LocatedIn` in Figure 3.3). These predictions can then be fed into the frontend of GeoDeepDive (Figure 3.4).

3.3 Scaling Feature Extraction

As shown in Figure 3.2, feature extraction is a key step in Elementary when processing a KBC task. To scale Elementary to web-scale KBC tasks, we employ high-throughput parallel computing frameworks such as Hadoop[6] and Condor[7] for feature extraction. We use the Hadoop File System for storage and MapReduce [26] for feature extraction. However, when trying to deploy Elementary to KBC tasks that involve terabytes of data and deep linguistic features, we found a 100-node Hadoop MapReduce cluster to be insufficient. The reasons are two-fold: (1) Hadoop's all-or-nothing approach to failure handling hinders throughput, and (2) the number of cluster machines for Hadoop is limited. Fortunately, the Condor infrastructure supports a best-effort failure model, i.e., a job may

[6] `http://hadoop.apache.org/`
[7] `http://research.cs.wisc.edu/condor/`

finish successfully even when Condor fails to process a small portion of the input data. Moreover, Condor can schedule jobs on both cluster machines and idle workstations. This allows us to simultaneously leverage thousands of machines from across a department, an entire campus, or even the nation-wide Open Science Grid.[8] For example, using Condor, we were able to utilize hundreds of thousands of machine-hours to perform deep linguistic feature extraction (including named entity recognition and dependency parsing) on the 500M-doc ClueWeb09 corpus[9] within a week.

3.4 Effectiveness of Statistical Inference

A main hypothesis of Elementary is that one can improve KBC result quality by combining more signals while resolving inconsistencies among these signals with statistical inference. Although the experimental results of several recent KBC projects have suggested that this is the case [17, 57, 84, 149], we use several more datasets to validate this hypothesis with Elementary. Specifically, we use Elementary to implement solutions to six different KBC tasks and measure how the result quality changes as we vary the the amount of signals (in the form of input data resources or MLN rules) in each solution. We find that overall more signals do tend to lead to higher quality in KBC tasks.

Datasets and MLNs We consider six KBC tasks:

1. **TAC**, which is the TAC-KBP (knowledge-base population)[10] challenge of populating a KB with 34 relations from a 1.8M-doc corpus by performing two related tasks: a) *entity linking*: extract all entity mentions and map them to entries in Wikipedia, and b) *slot filling*: determine relationships between entities. The MLN combines

[8] `http://www.opensciencegrid.org`
[9] `http://lemurproject.org/clueweb09.php/`
[10] `http://nlp.cs.qc.cuny.edu/kbp/2010/`

signals from Stanford NER[11], dependency paths from the Ensemble parser [127], web search results from Bing[12] (querying mention strings), and developers' feedback on linguistic patterns (in the form of MLN rules). For relation extraction, we perform distant supervision with Freebase as the training KB; Freebase is disjoint from the TAC-KBP benchmark ground truth. ELEMENTARY's quality for TAC-KBP is comparable to the state of the art [53] – we achieved an F1 score of 0.80 on entity linking (the best score in KBP2010 was 0.82; human performance is about 0.90) and 0.31 on slot filling (the best score in KBP2010 was 0.65, but all the other teams were lower than 0.30).

2. **CIA**, where the task is to populate the ternary person-title-in-country (PTIC) relation by processing the TAC corpus. The MLN combines signals such as a logistic regression model learned from distant supervision on linguistic patterns between entity mentions (including word sequences and dependency paths), name variations for persons, titles, and countries, and developer's feedback on the linguistic patterns. The ground truth is from the CIA World Factbook.[13] Following recent literature on distant supervision [50, 82, 147], we randomly split the World Factbook KB into equal-sized training set and testing set, and perform distant supervision with the training set to obtain a relation-extraction model. We measure output quality on the testing set portion of the KB. We will also report results after swapping the training and testing sets.

3. **IMDB**, where the task is to extract movie-director and movie-actor relationships from the TAC corpus. The MLN combines signals such as a logistic regression-based distant supervision model using lin-

[11] `http://nlp.stanford.edu/software/index.shtml`
[12] `http://www.bing.com/toolbox/bingdeveloper/`
[13] `https://www.cia.gov/library/publications/the-world-factbook/`

guistic patterns between entity mentions (including word sequences and dependency paths), a manually crafted list of erroneous movie names, and developer's feedback on linguistic patterns. The ground truth is from the IMDB dataset.[14] For relation-extraction model training and quality evaluation, we follow the same methodology as in CIA.

4. **NFL**, where the task is to extract National Football League game results (winners and losers) in the 2006-07 season from about 1.1K sports news articles. The MLN combines signals from a CRF-based team mention extractor, a dictionary of NFL team-name variations, and domain knowledge such as "a team cannot be both a winner and a loser on the same day." The CRF component is trained on game results from another season. The weights of the other rules are heuristically set. We use the actual game results as ground truth.[15]

5. **Enron**, where the task is to identify person mentions and associated phone numbers from 680 emails in the Enron dataset.[16] The MLN is derived from domain knowledge used by a rule-based IE system [21, 75] – it combines signals such as a list of common person names and variations, regular expressions for phone numbers, email senders' names, and domain knowledge that "a person doesn't have many phone numbers." The rule weights are heuristically set; no weight learning is involved. We use our manual annotation of the 680 emails as ground truth.

6. **DBLife**, where the task is to extract persons, organizations, and affiliation relationships between them from a collection of academic webpages.[17] The MLN is derived from domain knowledge used by

[14] `http://www.imdb.com/interfaces`
[15] `http://www.pro-football-reference.com/`
[16] `http://www.cs.cmu.edu/~einat/datasets.html`
[17] `http://dblife.cs.wisc.edu`

	Base	Base+EL	Base+RE	Full
TAC	string-based EL, DS for RE	+web search	+domain knowledge	all
CIA	string-based EL, DS for RE	+name variations	+domain knowledge	all
IMDB	string-based EL, DS for RE	+erroneous names	+domain knowledge	all
NFL	CRF winner/loser labeling	N/A	+domain knowledge	all
Enron	person-phone CO heuristic	+person coref	+domain knowledge	all
DBLife	person-org CO heuristic	+name variations	+relaxed CO	all

Table 3.1: *Signals in the four versions of MLN programs for each KBC task. DS means "distant supervision"; EL means "entity linking"; RE means "relation extraction." CO means "co-occurrences."*

another rule-based IE system [28] – it combines signals such as person names and variations (e.g., first/last name only, titles), organization names and a string-based similarity metric, and several levels of person-organization co-occurrence (adjacent, same-line, adjacent-line, etc.). The rule weights are heuristically set; no weight learning is involved. We use the ACM author profile data as ground truth.[18]

Methodology For each of the above KBC tasks, we consider four versions of MLN programs with different amounts of signals (Table 3.1): **Base** is a baseline, **Base+EL** (resp. **Base+RE**) has enhanced entity-linking (EL) (resp. relation-extraction (RE)) signals, and **Full** has all signals. For each task, we run each version of MLN program with marginal inference on Elementary until convergence. We then take the output (sorted by marginal probabilities) and plot a precision-recall curve. In precision-recall curves, higher is better.

Results As shown in Figure 3.5, the overall result is that more signals do help improve KBC quality: on each task, the quality of either Base+EL or Base+RE improves upon Base, and the Full program performs the best among all four versions. Specifically, on the TAC task, enhancing

[18] `http://www.acm.org/membership/author_pages`

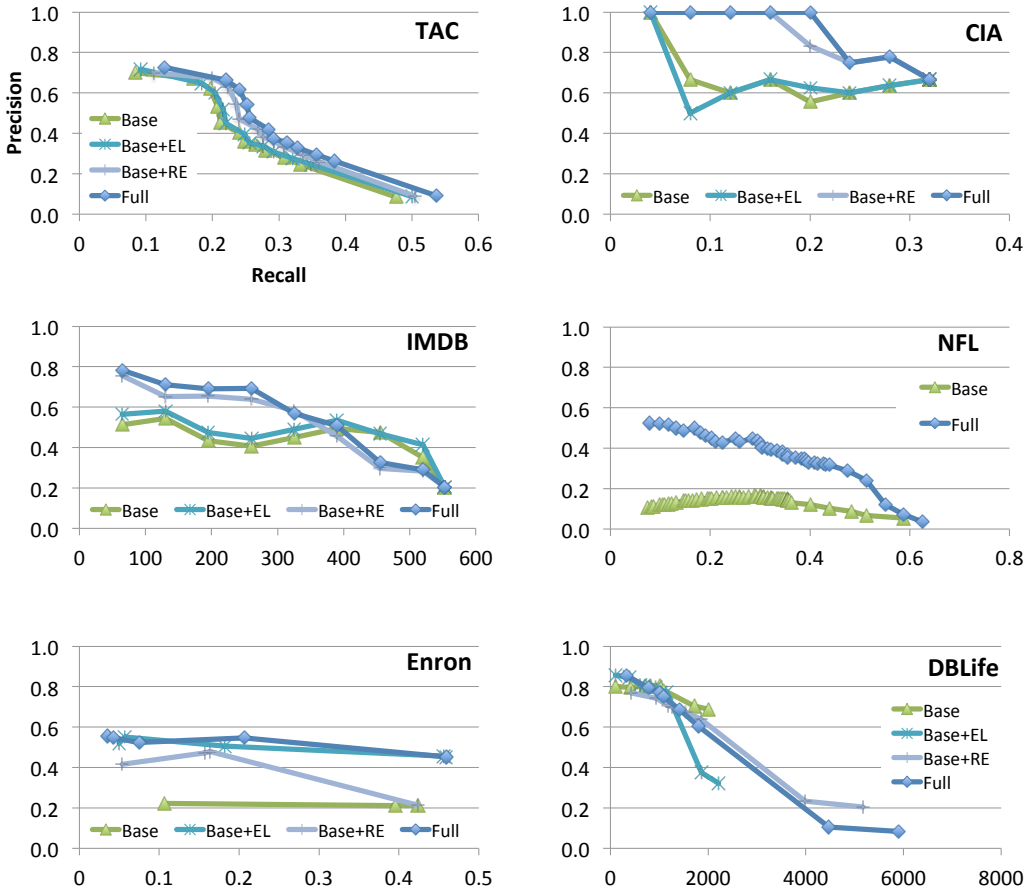


Figure 3.5: *Result quality of KBC tasks improves as we add more signals into* Elementary. *On each of the six tasks, we run four versions of* Elementary *programs and plot the precision-recall curves for each version:* ***Base*** *is the baseline version,* ***Base+EL*** *has enhanced signals for entity linking,* ***Base+RE*** *has enchanced signals for relation extraction, and* ***Full*** *has enhanced signals for both entity linking and relation extraction. [*] X-axes with integer labels correspond to datasets where we use the true positive count to measure recall; the reason is that the corpus covers only a small portion of the ground-truth KB. NFL has only two curves because we do not have enhanced EL signals.*

either EL or RE signals improves both precision and recall; futhermore, the combination of such enhancements (resulting in the Full program)

achieves even higher quality than all the other three settings. These results suggest that integrating signals with statistical inference is a promising approach to KBC challenges such as TAC-KBP. On the single-relation KBC tasks CIA, NFL, and Enron, we observe even more significant quality improvement as more signals are used by Elementary: on each dataset, the Full program has almost twice as high precision at each recall level compared to the Base program.

On IMDB, the addition of enhanced EL signals slightly but consistently improves precision (Base+EL over Base, and Full over Base+RE); the addition of enhanced RE signals effectively helps segregate high-confident results from low-confident results (the slope of the curve becomes steeper). On DBLife, the enhanced EL or RE signals help improve the precision of the most-confident results (the first 500 predictions or so), while extending the largest recall value substantially. From the above results, we conclude that one can improve the quality of KBC tasks by integrating diverse data and signals via statistical inference. To further verify the above observations, we also swap the training and testing sets for IMDB and CIA. On either IMDB or CIA, the obtained essentially the same results as in Figure 3.5.

4 Scaling Distant Supervision for KBC

A key challenge to web-scale KBC is the ability to handle the variety of how an entity or a relationship is expressed (i.e., *mentioned*) in natural-language text. To deal with this challenge, researchers apply machine learning techniques in KBC. A key issue for machine learning is the availability of training data, and an increasingly popular technique addressing this issue is *distant supervision* that generates training examples from scalable data souces such as text corpora, existing knowledge bases, and crowdsourcing. To guide the development of web-scale KBC systems such as Elementary, we study how scaling the input to distant supervision impacts KBC quality. Most of the work in this chapter appears in Zhang, Niu, Ré, and Shavlik [148].

4.1 Motivations

Relation extraction is the problem of populating a *target relation* (representing an entity-level relationship or attribute) with facts extracted from natural-language text. Sample relations include people's titles, birth places, and marriage relationships.

Traditional relation-extraction systems rely on manual annotations or domain-specific rules provided by experts, both of which are scarce resources that are not portable across domains. To remedy these problems, recent years have seen interest in the *distant supervision* approach for relation extraction [82, 144]. The input to distant supervision is a set of *seed facts* for the target relation together with an (unlabeled) text corpus, and the output is a set of (noisy) annotations that can be used by any machine learning technique to train a statistical model for the target relation. For example, given the target relation `birthPlace`(person, place) and a seed fact `birthPlace`(John, Springfield), the sentence "*John and his wife were born*

in Springfield in 1946" (S1) would qualify as a positive training example.

Distant supervision replaces the expensive process of manually acquiring annotations that is required by direct supervision with resources that already exist in many scenarios (seed facts and a text corpus). On the other hand, distantly labeled data may not be as accurate as manual annotations. For example, "*John left Springfield when he was 16*" (S2) would also be considered a positive example about place of birth by distant supervision as it contains both John and Springfield. The hypothesis is that the broad coverage and high redundancy in a large corpus would compensate for this noise. For example, with a large enough corpus, a distant supervision system may find that patterns in the sentence S1 strongly correlate with seed facts of `birthPlace` whereas patterns in S2 do not qualify as a strong indicator. Thus, intuitively the quality of distant supervision should improve as we use larger corpora. However, there has been no study on the impact of corpus size on distant supervision for relation extraction. Our goal is to fill this gap.

Besides "big data," another resource that may be valuable to distant supervision is crowdsourcing. For example, one could employ crowd workers to provide feedback on whether distant supervision examples are correct or not [43]. Intuitively the crowd workforce is a perfect fit for such tasks since many erroneous distant labels could be easily identified and corrected by humans. For example, distant supervision may mistakenly consider "*Obama took a vacation in Hawaii*" a positive example for `birthPlace` simply because a database says that Obama was born in Hawaii; a crowd worker would correctly point out that this sentence is not actually indicative of this relation.

It is unclear however which strategy one should use: scaling the text corpus or the amount of human feedback. Our primary contribution is to empirically assess how scaling these inputs to distant supervision impacts its result quality. We study this question with input data sets

that are orders of magnitude larger than those in prior work. While the largest corpus (Wikipedia and New York Times) employed by recent work on distant supervision [50, 82, 147] contain about 2M documents, we run experiments on a 100M-document (50X more) corpus drawn from ClueWeb.[1] While prior work [43] on crowdsourcing for distant supervision used thousands of human feedback units, we acquire tens of thousands of human-provided labels. Despite the large scale, we follow state-of-the-art distant supervision approaches and use deep linguistic features, e.g., part-of-speech tags and dependency parsing.[2]

Our experiments shed insight on the following two questions:

1. *How does increasing the corpus size impact the quality of distant supervision?*

2. *For a given corpus size, how does increasing the amount of human feedback impact the quality of distant supervision?*

We find that increasing corpus size consistently and significantly improves recall and F1, despite reducing precision on small corpora; in contrast, human feedback has relatively small impact on precision and recall. For example, on a TAC corpus with 1.8M documents, we found that increasing the corpus size ten-fold consistently results in statistically significant improvement in F1 on two standardized relation extraction metrics (t-test with p=0.05). On the other hand, increasing human feedback amount ten-fold results in statistically significant improvement on F1 only when the corpus contains at least 1M documents; and the magnitude of such improvement was only one fifth compared to the impact of corpus-size increment.

We find that the quality of distant supervision tends to be *recall gated*, that is, for any given relation, distant supervision fails to find all possible

[1] `http://lemurproject.org/clueweb09.php/`

[2] We used 100K CPU hours to run such tools on ClueWeb.

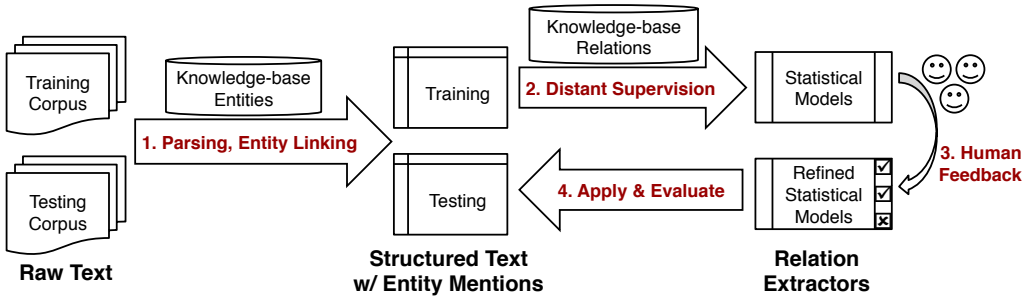


Figure 4.1: *The workflow of our distant supervision system. Step 1 is preprocessing; step 4 is final evaluation. The key steps are distant supervision (step 2), where we train a logistic regression (LR) classifier for each relation using (noisy) examples obtained from sentences that match Freebase facts, and human feedback (step 3) where a crowd workforce refines the LR classifiers by providing feedback to the training data.*

linguistic signals that indicate a relation. By expanding the corpus one can expand the number of patterns that occur with a known set of entities. Thus, as a rule of thumb for developing distant supervision systems, one should first attempt to expand the training corpus and then worry about precision of labels only after having obtained a broad-coverage corpus.

Throughout this chapter, it is important to understand the difference between *mentions* and *entities*. Entities are conceptual objects that exist in the world (e.g., Barack Obama), whereas authors use a variety of wordings to refer to (which we call "mention") entities in text [53].

4.2 Distant Supervision Methodology

Relation extraction is the task of identifying relationships between mentions, in natural-language text, of entities. An example relation is that two persons are married, which for mentions of entities x and y is denoted $R(x, y)$. Given a corpus C containing mentions of named entities, our goal is to learn a classifier for $R(x, y)$ using linguistic features of x

and y, e.g., dependency-path information. The problem is that we lack the large amount of labeled examples that are typically required to apply supervised learning techniques. We describe an overview of these techniques and the methodological choices we made to implement our study. Figure 4.1 illustrates the overall workflow of a distant supervision system. At each step of the distant supervision process, we closely follow the recent literature [82, 147].

4.2.1 Distant Supervision

Distant supervision compensates for a lack of training examples by generating what are known as *silver-standard examples* [144]. The observation is that we are often able to obtain a structured, but incomplete, database D that instantiates relations of interest and a text corpus C that contains mentions of the entities in our database. Formally, a database is a tuple $D = (E, \bar{R})$ where E is a set of entities and $\bar{R} = (R_1 \ldots, R_N)$ is a tuple of instantiated predicates. For example, R_i may contain pairs of married people.[3] We use the facts in R_i combined with C to generate examples.

Following recent work [50, 82, 147], we use Freebase[4] as the knowledge base for seed facts. We use two text corpora: (1) the TAC-KBP[5] 2010 corpus that consists of 1.8M newswire and blog articles[6], and (2) the ClueWeb09 corpus that is a 2009 snapshot of 500M webpages. We use the TAC-KBP slot filling task and select those TAC-KBP relations that are present in the Freebase schema as targets (20 relations on people and organization).

One problem is that relations in D are defined at the entity level. Thus, the pairs in such relations are not embedded in text, and so these pairs lack the linguistic context that we need to extract features, i.e., the features used to describe examples. In turn, this implies that these pairs cannot be

[3] We only consider binary predicates in this work.

[4] `http://freebase.com`

[5] KBP stands for "Knowledge-Base Population."

[6] `http://nlp.cs.qc.cuny.edu/kbp/2010/`

used directly as training examples for our classifier. To generate training examples, we need to map the entities back to mentions in the corpus. We denote the relation that describes this mapping as the relation $\mathrm{EL}(e, m)$ where $e \in E$ is an entity in the database D and m is a mention in the corpus C. For each relation R_i, we generate a set of (noisy) positive examples denoted R_i^+ defined as $R_i^+ =$

$$\{(m_1, m_2) \mid R(e_1, e_2) \wedge \mathrm{EL}(e_1, m_1) \wedge \mathrm{EL}(e_2, m_2)\}$$

As in previous work, we impose the constraint that both mentions $(m_1, m_2) \in R_i^+$ are contained in the same sentence [50, 82, 147]. To generate negative examples for each relation, we follow the assumption in Mintz et al. [82] that relations are disjoint and sample from other relations, i.e., $R_i^- = \cup_{j \neq i} R_j^+$.

Relations We select the target relations by intersecting the TAC-KBP relations and the Freebase schema, resulting in a list of 20 relations as shown in Figure 4.2. These relations are fairly commonplace in relation extraction tasks, and they cover all the binary MR-KBP relations[7].

4.2.2 Feature Extraction

Once we have constructed the set of possible mention pairs, the state-of-the-art technique to generate feature vectors uses linguistic tools such as part-of-speech taggers, named-entity recognizers, dependency parsers, and string features. Following recent work on distant supervision [50, 82, 147], we use both lexical and syntactic features. After this stage, we have a well-defined machine learning problem that is solvable using standard supervised techniques. We use *sparse logistic regression* (ℓ_1 regularized) [134], which is used in previous studies. Our feature extraction process consists of three steps:

[7] MR stands for Machine Reading, a DARPA project, and MR-KBP is a test case/benchmark in the project and is similar to TAC-KBP.

Subject	Relation	# Facts
person	date_of_birth	887,079
person	title	454,707
person	city_of_birth	354,891
organization	subsidiaries	221,835
organization	parents	219,542
person	schools_attended	165,232
organization	city_of_headquarters	157,839
person	cities_of_residence	122,610
person	city_of_death	97,376
person	employee_of	86,381
person	spouse	34,796
person	parents	33,598
person	children	33,598
person	state_of_residence	31,494
person	siblings	16,610
organization	top_members	8,307
organization	founded_by	8,294
person	countries_of_residence	8,183
organization	number_of_employees	2,610
person	member_of	2,290

Figure 4.2: *Relations used in the experiments and the number of corresponding facts in Freebase. We select these relations by taking all TAC-KBP 2010 target relations that are present in Freebase. These relations cover all (binary) MR-KBP relations.*

1. Run Stanford CoreNLP with POS tagging and named entity recognition [40];

2. Run dependency parsing on TAC with the Ensemble parser [127] and on ClueWeb with MaltParser[8] [91]; and

[8]We did not run Ensemble on ClueWeb because we had very few machines satisfying Ensemble's memory requirement. In contrast, MaltParser requires less memory and we could leverage Condor [132] to parse ClueWeb with MaltParser within several days (using about 50K CPU hours).

3. Run a simple entity-linking system that utilizes NER results and string matching to identify mentions of Freebase entities (with types).[9]

The output of this processing is a repository of structured objects (with POS tags, dependency parse, and entity types and mentions) for sentences from the training corpus. Specifically, for each pair of entity mentions (m_1, m_2) in a sentence, we extract the following features $F(m_1, m_2)$: (1) the word sequence (including POS tags) between these mentions after normalizing entity mentions (e.g., replacing "John Nolen" with a place holder PER); if the sequence is longer than 6, we take the 3-word prefix and the 3-word suffix; (2) the dependency path between the mention pair. To normalize, in both features we use lemmas instead of surface forms. We discard features that occur in fewer than three mention pairs.

4.2.3 Crowd-Sourced Data

Crowd sourcing provides a cheap source of human labeling to improve the quality of our classifier. In this work, we specifically examine feedback on the result of distant supervision. Precisely, we construct the union of $R_1^+ \cup \ldots R_N^+$ from Section 4.2.1. We then solicit human labeling from Mechanical Turk (MTurk) while applying state-of-the-art quality control protocols following Gormley et al. [43] and those in the MTurk manual.[10]

These quality-control protocols are critical to ensure high quality: spamming is common on MTurk and some turkers may not be as proficient or careful as expected. To combat this, we replicate each question three times and, following Gormley et al. [43], plant gold-standard questions: each task consists of five yes/no questions, one of which comes from our

[9] We experiment with a slightly more sophisticated entity-linking system as well, which resulted in higher overall quality. The results below are from the simple entity-linking system.

[10] `http://mturkpublic.s3.amazonaws.com/docs/MTURK_BP.pdf`

gold-standard pool.[11] By retaining only those answers that are consistent with this protocol, we are able to filter responses that were not answered with care or competency. We only use answers from workers who display overall high consistency with the gold standard (i.e., correctly answering at least 80% of the gold-standard questions).

MTurk Question Formulation For each selected example, we replace the corresponding entity mentions in the sentence with entity-type place holders[12], and then submit the resultant sentence with a yes/no question to the crowd for an answer. An example question is

> Does the following sentence literally imply that PERSON1 is PERSON2's spouse? *... PERSON1 – who was pregnant with PERSON2 ...*

4.2.4 Statistical Modeling

Following Mintz et al. [82], we use logistic regression classifiers to represent relation extractors. However, while Mintz et al. use a single multi-class classifier for all relations, Hoffman et al. [50] and use an independent binary classifier for each individual relation; the intuition is that a pair of mentions (or entities) might participate in multiple target relations. We experimented with both protocols; since relation overlapping is rare for TAC-KBP and there was little difference in result quality, we focus on the binary-classification approach using training examples constructed as described in Section 4.2.1.

We compensate for the different sizes of distant and human labeled examples by training an objective function that allows to tune the weight

[11] We obtain the gold standard from a separate MTurk submission by taking examples that at least 10 out of 11 turkers answered yes, and then negate half of these examples by altering the relation names (e.g., `spouse` to `sibling`).

[12] We empirically found that showing entity types instead of concrete entity names helps reduce crowd workers' tendency of answering based on background knowledge.

of human versus distant labeling. We separately tune this parameter for each training set (with cross validation), but found that the result quality was robust with respect to a broad range of parameter values.

We represent a set of training examples as $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ where each $\mathbf{x}_i$ is a Boolean vector representing the features of example i and $y_i = 1$ if it is a positive example and -1 otherwise. Then the objective function for training the classifier of a relation r is

$$f(\mathbf{w}) = \sum_{i=1}^{m} \log(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i)) + \mu \|\mathbf{w}\|_1,$$

where $\mathbf{w}$ is the weight vector representing the classifier and μ is a regularization parameter that controls the trade-off between the model complexity and training errors. We use ℓ_1-norm regularization since the models are expected to be sparse [134]. We use a held-out portion of distant labels to tune μ, found that the model is relatively insensitive to the value of μ, and so selected a reasonable $\mu = 0.001m$ for our experiments.

To incorporate human feedback on distant-labeled examples, we augment the above objective function as follows. Suppose there are n units of human feedback, and denote by z_j the value (1 if positive, -1 if negative) of feedback j on example i_j. To account for the intuition that human labels should weigh differently than distant labels, we use the following objective function:

$$\begin{aligned} g(\mathbf{w}) &= (1-\nu) \sum_{i=1}^{m} \log(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i)) \\ &+ \nu \sum_{j=1}^{n} \log(1 + \exp(-z_j \mathbf{w}^\top \mathbf{x}_{i_j})) \\ &+ \mu \|\mathbf{w}\|_1, \end{aligned}$$

where $\nu \in [0, 1]$ intuitively controls how much we weigh human feedback relative to the original distant-supervised labels. We use a held-out portion of distantly supervised training data (on a held-out portion of the training corpus) to tune the parameters ν for different input sizes, but found that the optimal value for ν is virtually always the minimum value in the search space [13]. Thus, we heuristically set $\nu = 0.8$ for our experiments. We report sensitivity study results in Section 4.3.7.

4.3 Experiments

We describe our experiments to test the hypotheses that the following two factors improve distant-supervision quality: increasing the

(1) corpus size, and

(2) the amount of crowd-sourced feedback.

We confirm hypothesis (1), but, surprisingly, are unable to confirm (2). Specifically, when using logistic regression to train relation extractors, increasing corpus size improves, consistently and significantly, the precision and recall produced by distant supervision, regardless of human feedback levels. Using the methodology described in Section 4.2, human feedback has limited impact on the precision and recall produced from distant supervision by itself.

4.3.1 Evaluation Metrics

Just as direct training data are scarce, ground truth for relation extraction is scarce as well. As a result, prior work mainly considers two types of evaluation methods: (1) randomly sample a small portion of predictions

[13] We conjecture that this is because distant labels and human labels have different biases and the validation set always leans toward distant labels. More evidence in Section 4.3.6.

(e.g., top-k) and manually evaluate precision/recall; and (2) use a held-out portion of seed facts (usually Freebase) as a kind of "distant" ground truth. We replace manual evaluation with a standardized relation-extraction benchmark: TAC-KBP 2010. TAC-KBP asks for extractions of 46 relations on a given set of 100 entities. Interestingly, the Freebase held-out metric [50, 82, 147] turns out to be heavily biased toward distantly labeled data (e.g., increasing human feedback *hurts* precision; see Section 4.3.6).

4.3.2 Experimental Setup

Our first group of experiments use the 1.8M-doc TAC-KBP corpus for training. We exclude from it the 33K documents that contain query entities in the TAC-KBP metrics. There are two key parameters: the corpus size (#docs) M and human feedback budget (#examples) N. We perform different levels of down-sampling on the training corpus. On TAC, we use subsets with $M = 10^3, 10^4, 10^5$, and 10^6 documents respectively. For each value of M, we perform 30 independent trials of uniform sampling, with each trial resulting in a training corpus D_i^M, $1 \leqslant i \leqslant 30$. For each training corpus D_i^M, we perform distant supervision to train a set of logistic regression classifiers. From the full corpus, distant supervision creates around 72K training examples.

To evaluate the impact of human feedback, we randomly sample 20K examples from the input corpus (we remove any portion of the corpus that is used in an evaluation). Then, we ask three different crowd workers to label each example as either positive or negative using the procedure described in Section 4.2.3. We retain only credible answers using the gold-standard method (see Section 4.2.3), and use them as the pool of human feedback that we run experiments with. About 46% of our human labels are negative. Denote by N the number of examples that we want to incorporate human feedback for; we vary N in the range of $0, 10, 10^2, 10^3, 10^4$, and 2×10^4. For each selected corpus and value of N, we perform without-

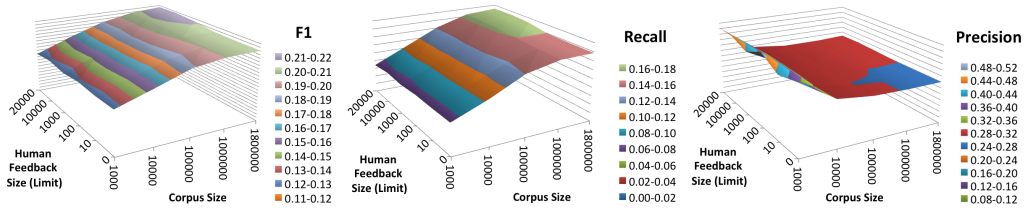


Figure 4.3: *Impact of input sizes under the TAC-KBP metric, which uses documents mentioning 100 predefined entities as testing corpus with entity-level ground truth. We vary the sizes of the training corpus and human feedback while measuring the scores (F1, recall, and precision) on the TAC-KBP benchmark.*

replacement sampling from examples of this corpus to select feedback for up to N examples. In our experiments, we found that on average an M-doc corpus contains about 0.04M distant labels, out of which 0.01M have human feedback. After incorporating human feedback, we evaluate the relation extractors on the TAC-KBP benchmark as well as a MR-KBP benchmark.[14] We then compute the average F1, recall, and precision scores among all trials for each metric and each (M,N) pair.

Besides the KBP metrics, we also evaluate each (M,N) pair using Freebase held-out data. Furthermore, we experiment with a much larger corpus: ClueWeb09. On ClueWeb09, we vary M over $10^3, \ldots, 10^8$. Using the same metrics, we show at a larger scale that increasing corpus size can significantly improve both precision and recall.

4.3.3 Overall Impact of Input Sizes

We first present our experiment results on the TAC corpus. Results from the TAC-KBP metric (Figure 4.3) and the MR-KBP metric (Figure 4.4) are similar. In both cases, the F1 graph closely tracks the recall graph, which

[14]The MR-KBP metric was originally for temporal relation extraction and consists of a set of ground truth annotations in 75 testing documents. For our experiments, we discard the temporal aspects of the annotations.

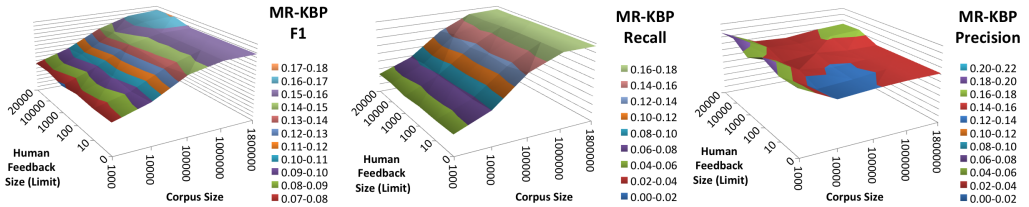


Figure 4.4: *Impact of input sizes under the MR-KBP metric, which has a 75-doc testing corpus and mention-level ground truth. The result is very similar to TAC-KBP.*

	$M = 10^3$	$M = 1.8 \times 10^6$
$N = 0$	0.124	0.201
$N = 2 \times 10^4$	0.118	0.214

Table 4.1: *TAC F1 scores with max/min values of* M/N.

supports our earlier claim that quality is recall gated (Section 6.1). We analyze the TAC-KBP results below. While increasing the corpus size improves F1 at a roughly log-linear rate, human feedback has little impact until both corpus size and human feedback size approch maximum M, N values. Table 4.1 shows the quality comparisons with minimum/maximum values of M and N.[15] We observe that increasing the corpus size significant improves per-relation recall and F1 on 17 out of TAC-KBP's 20 relations; in contrast, human feedback has little impact on recall, and only significantly improves the precision and F1 of 9 relations – while hurting F1 of 2 relations (i.e., `MemberOf` and `LivesInCountry`).

[15]When the corpus size is small, the total number of examples with feedback can be smaller than the budget size N – for example, when $M = 10^3$ there are on average 10 examples with feedback even if $N = 10^4$.

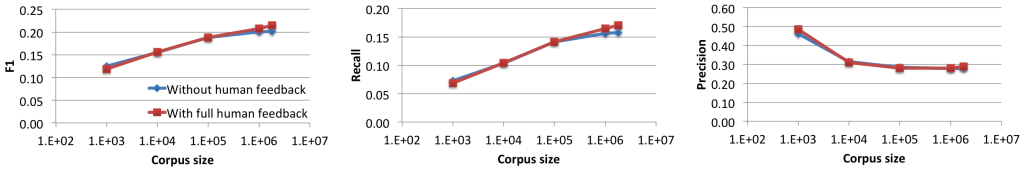


Figure 4.5: *Projections of Figure 4.3 to show the impact of corpus size changes under the TAC-KBP metric.*

M\N	0	10	10^2	10^3	10^4	2e4
$10^3 \rightarrow 10^4$	+	+	+	+	+	+
$10^4 \rightarrow 10^5$	+	+	+	+	+	+
$10^5 \rightarrow 10^6$	+	+	+	+	+	+
$10^6 \rightarrow$ 1.8e6	0	0	0	+	+	+

Table 4.2: *Two-tail t-test with d.f.=29 and p=0.05 on the impact of corpus size. (We also tried p=0.01, which resulted in change of only a single cell.) Each column corresponds to a fixed human-feedback budget size* N*. Each row corresponds to a jump from one corpus size (*M*) to the immediate larger size. Each cell value indicates whether the TAC-KBP F1 metric changed significantly: + (resp. -) indicates that the quality increased (resp. decreased) significantly; 0 indicates that the quality did not change significantly.*

4.3.4 Impact of Corpus Size

In Figure 4.5 we plot a projection of the graphs in Figure 4.3 to show the impact of corpus size on distant-supervision quality. The two curves correspond to when there is no human feedback and when we use all applicable human feedback. The fact that the two curves almost overlap indicates that human feedback had little impact on precision or recall. On the other hand, the quality improvement rate is roughly log-linear against the corpus size. Recall that each data point in Figure 4.3 is the average from 30 trials. To measure the statistical significance of changes in F1, we calculate t-test results to compare adjacent corpus size levels given each fixed human feedback level. As shown in Table 4.2, increasing the

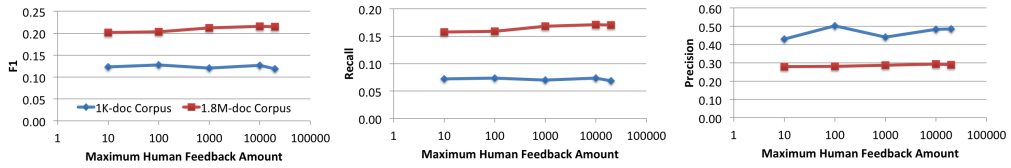


Figure 4.6: *Projections of Figure 4.3 to show the impact of human feedback amount under the TAC-KBP metric.*

N\M	10^3	10^4	10^5	10^6	1.8e6
$0 \rightarrow 10$	0	0	0	0	0
$10 \rightarrow 10^2$	0	0	0	+	+
$10^2 \rightarrow 10^3$	0	0	0	+	+
$10^3 \rightarrow 10^4$	0	0	0	0	+
$10^4 \rightarrow$ 2e4	0	0	0	0	-
$0 \rightarrow$ 2e4	0	0	0	+	+

Table 4.3: *Two-tail t-test with d.f.=29 and p=0.05 on the impact of human feedback size changes. (We also tried p=0.01 but obtained the same result.) Each column corresponds to a fixed corpus size* M*. Each row corresponds to a jump from one feedback size (*N*) to the immediate larger size. Each cell value indicates whether the TAC F1 metric changed significantly: + (resp. -) indicates that the quality increased (resp. decreased) significantly; 0 indicates that the quality did not change significantly.*

corpus size by a factor of 10 consistently and significantly improves F1. Although precision decreases as we use larger corpora, the decreasing trend is sub-log-linear and stops at around 100K docs. On the other hand, recall and F1 keep increasing at a log-linear rate.

4.3.5 Impact of Human Feedback

Figure 4.6 provides another perspective on the results under the TAC metric: We fix a corpus size and plot the F1, recall, and precision as functions of human-feedback amount. Confirming the trend in Figure 4.3,

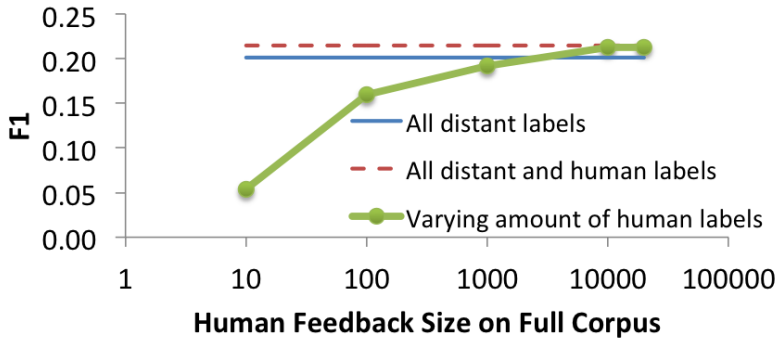


Figure 4.7: *TAC-KBP quality of relation extractors trained using different amounts of human labels. The horizontal lines are comparison points.*

we see that human feedback has little impact on precision or recall with both corpus sizes.

We calculate t-tests to compare adjacent human feedback levels given each fixed corpus size level. Table 4.3's last row reports the comparison, for various corpus sizes (and, hence, number of distant labels), of (i) using no human feedback and (ii) using *all* of the human feedback we collected. When the corpus size is small (fewer than 10^5 docs), human feedback has no statistically significant impact on F1. The locations of $+$'s suggest that the influence of human feedback becomes notable only when the corpus is very large (say with 10^6 docs). However, comparing the slopes of the curves in Figure 4.6 against Figure 4.5, the impact of human feedback is substantially smaller. The precision graph in Figure 4.6 suggests that human feedback does not notably improve precision on either the full corpus or on a small 1K-doc corpus.

To assess the quality of human labels, we train extraction models with human labels only (on examples obtained from distant supervision). We vary the amount of human labels and plot the F1 changes in Figure 4.7. Although the F1 improves as we use more human labels, the best model has roughly the same performance as those trained from distant labels

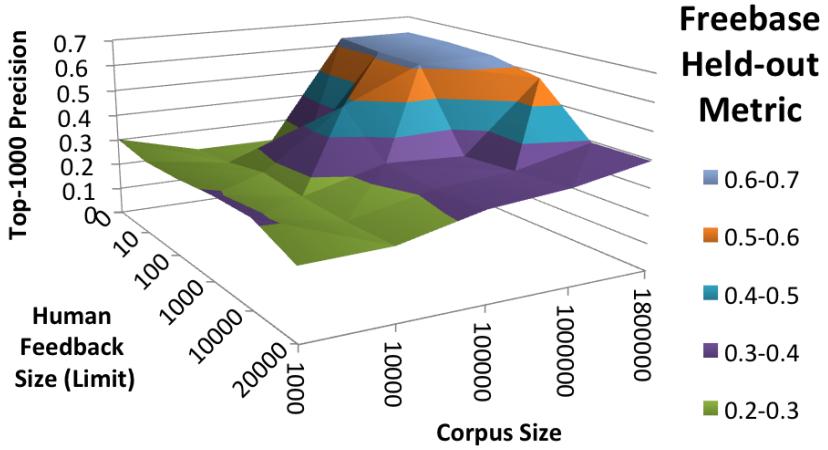


Figure 4.8: *Impact of input sizes under the Freebase held-out metric. Note that the human feedback axis is in the reverse order compared to Figure 4.3.*

(with or without human labels). This suggests that the accuracy of human labels is not substantially better than distant labels.

4.3.6 Freebase Held-out Metric

In addition to the TAC-KBP benchmark, we also follow prior work [50, 82, 147] and measure the quality using held-out data from Freebase. We randomly partition both Freebase and the corpus into two halves. One database-corpus pair is used for training and the other pair for testing. We evaluate the precision over the 10^3 highest-probability predictions on the test set. In Figure 4.8, we vary the size of the corpus in the train pair and the number of human labels; the precision reaches a dramatic peak when we the corpus size is above 10^5 *and* uses little human feedback. This suggests that this Freebase held-out metric is biased toward solely relying on distant labels alone.

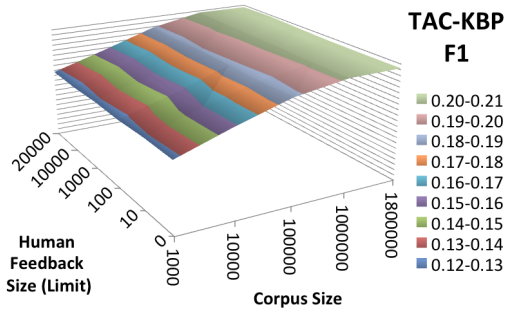


Figure 4.9: *Impact of input sizes with* $\nu = 0.5$ *under the TAC-KBP metric.*

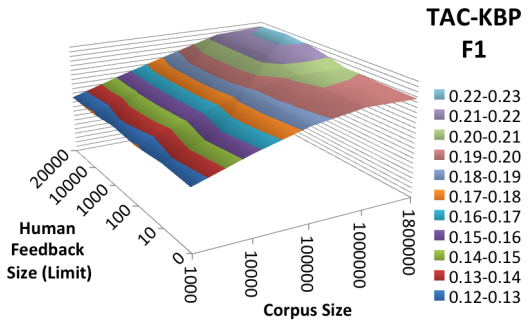


Figure 4.10: *Impact of input sizes with* $\nu = 0.9$ *under the TAC-KBP metric.*

4.3.7 Sensitivity Studies

In this section, we describe several experiments designed to validate the robustness of our findings.

First, we vary ν and see how experiment results change accordingly. For example, we compare $\nu = 0.5$ (Figure 4.9) and $\nu = 0.9$ (Figure 4.10) against the default value $\nu = 0.8$ (Figure 4.3). We see that ν seems to have a monotone impact on the final quality of full corpus with full human feedback: the higher ν is, the better the quality on the $\max(M, N)$ corner and the less moother the full corpus projection curve. Still, even with $\nu = 0.999$ (results not shown but very similar to $\nu = 0.9$), the slope of

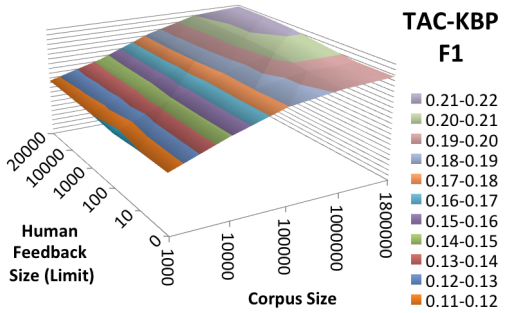


Figure 4.11: *Impact of input sizes with a 33%-down-sampled Freebase KB under the TAC-KBP metric.*

the full-corpus projection curve is much smaller than the zero-human projection curve.

In the second sensitivity study, we evaluate how changing the KB size (Freebase in this case) influences the result quality. In Figure 4.11, we use a down-sampled Freebase instance that is only a third the original size, and found that, even though at each grid point the number of examples is only a third as before (as expected), when the corpus is large enough, distant supervision still manages to retain essentially the same quality. This demonstrates how larger corpora achieve higher quality through higher redundancy and robustness.

4.3.8 Web-scale Corpora

The corpus for the TAC benchmark is crafted to contain mentions of entities in TAC in the test corpus. In contrast, the Web is less curated, but does contain mentions of many entities contained in TAC (and many other databases). To study how a Web corpus impacts distant-supervision quality, we select the first 100M English webpages from the ClueWeb09 dataset and measure how distant-supervision quality changes as we vary the number of webpages used.

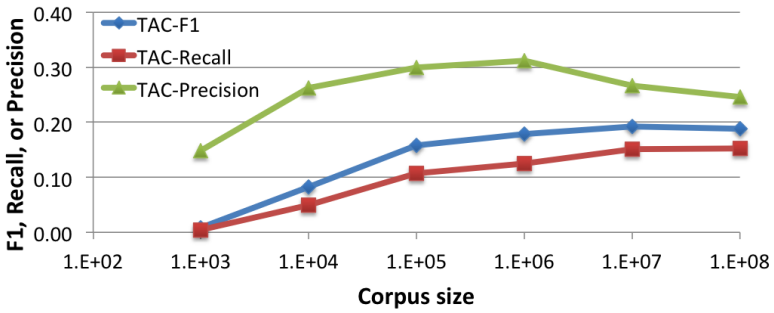


Figure 4.12: *Impact of corpus size on TAC-KBP with ClueWeb*

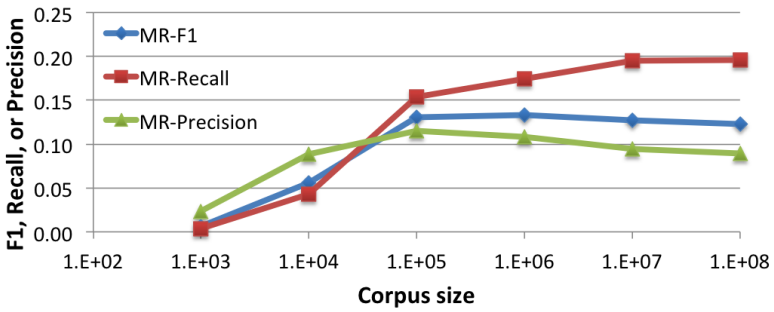


Figure 4.13: *Impact of corpus size on MR-KBP with ClueWeb*

In Figure 4.12 and Figure 4.13, we plot the TAC-KBP and MR-KBP quality graphs as a function of changing ClueWeb corpus sizes. Table 4.4 shows the statistical significance test results. For TAC-KBP, increasing the corpus size improves F1 up to 10^7 docs ($p = 0.05$), while at 10^8 the two-tailed significance test reports no significant impact on F1 ($p = 0.05$). The dip in precision in Figure 4.12 from 10^6 to either 10^7 or 10^8 is significant ($p = 0.05$), and it is interesting future work to perform a detailed error analysis. For MR-KBP, the corpus size similarly has a consistently positve impact on recall that converges roughly after 10^7 documents. Compared to TAC-KBP, the F1 score is more sensitive to decrease of precision and converges at around 10^6 documents and then slightly decreases. The

Corpus Size	TAC-KBP F1	MR-KBP F1
$10^3 \rightarrow 10^4$	+	+
$10^4 \rightarrow 10^5$	+	+
$10^5 \rightarrow 10^6$	+	0
$10^6 \rightarrow 10^7$	+	0
$10^7 \rightarrow 10^8$	0	-

Table 4.4: *Two-tail t-test with d.f.=9 and p=0.05 on the impact of corpus size and feedback size changes respectively. Each row corresponds to a jump from one corpus size (M) to the immediate larger size. Each cell value indicates whether the TAC or MR F1 metric changed significantly: + (resp. -) indicates that the quality increased (resp. decreased) significantly; 0 indicates that the quality did not change significantly.*

reason might be that the MR-KBP test corpus is substantially smaller than TAC-KBP (75 docs vs. 33K docs).

Recall from Section 4.2 that to preprocess ClueWeb we use MaltParser instead of Ensemble. Thus, the F1 scores in Figure 4.12 Figure 4.13 are not comparable to those from the TAC training corpus.

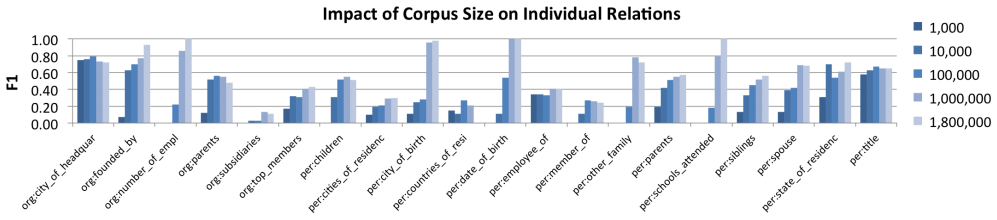


Figure 4.14: *Impact of corpus size on individual relations vary between relations. We vary the training corpus size and measure the quality numbers of individual relations. The F1 score of each relation is based on the extraction results for the TAC-KBP slot filling benchmark. The "per" prefix of relation names means that the subject of this relation is a person; similarly "org" is for organization.*

4.3.9 Per-relation Analysis

In this section, we dig deeper to analyze the impact of coprus sizes on the extraction quality of individual target relations with distant supervision. As shown in Figure 4.14, different relations have different sensitivities to the increase of corpus sizes. Overall, the qualities of most relations (e.g., `spouse`, `siblings`, `city_of_birth`, and `school_attended`) improve significantly as we increase the corpus size. Other relations (e.g., `member_of` and `employee_of`) are less sensitive.

4.4 Discussion

We study how the size of two types of cheaply available resources impact the precision and recall of distant supervision: (1) an unlabeled text corpus from which distantly labeled training examples can be extracted, and (2) crowd-sourced labels on training examples. We found that text corpus size has a stronger impact on precision and recall than human feedback. We observed that distant-supervision systems are often *recall gated*; thus, to improve distant-supervision quality, one should first try to enlarge the input training corpus and then increase precision.

It was initially counter-intuitive to us that human labels did not have a large impact on precision. One reason is that human labels acquired from crowd-sourcing have comparable noise level as distant labels – as shown by Figure 4.7. Thus, techniques that improve the accuracy of crowd-sourced answers are an interesting direction for future work. We used a particular form of human input (yes/no votes on distant labels) and a particular statistical model to incorporate this information (logistic regression). It is interesting future work to study other types of human input (e.g., new examples or features) and more sophisticated techniques for incorporating human input, as well as machine learning methods that explicitly model feature interactions.

5 Scaling Markov Logic using an RDBMS

The motivating observation of this chapter is that a crucial step in Markov logic inference, called *grounding*, essentially consists of relational operations like SQL. Thus, instead of performing grounding in main memory and with ad hoc optimizations – as in prior Markov logic systems – we study how to leverage the scalability and efficiency of an RDBMS for Makov logic inference. The work in this chapter appears in Niu, Ré, Doan, and Shavlik [87].

5.1 Motivations, Challenges, and Contributions

Over the past few years, Markov Logic Networks (MLNs) have emerged as a powerful and popular framework combining logical and probabilistic reasoning. MLNs have been successfully applied to a wide variety of data management problems, e.g., information extraction, entity resolution, and text mining. In contrast to probability models like factor graphs [112] that require complex distributions to be specified in tedious detail, MLNs allow us to declare a rigorous statistical model at a much higher conceptual level using first-order logic. For example, to classify papers by research area, one could write a rule such as *"it is likely that if one paper cites another they are in the same research area."*

Our interest in MLNs stems from our involvement in a DARPA project called "Machine Reading." The grand challenge is to build software that can read the Web, i.e., extract and integrate structured data (e.g., entities, relationships) from Web data, then use this structured data to answer user queries. The current approach is to use MLNs as a lingua franca to combine many different kinds of extractions into one coherent picture. To accomplish this goal, it is critical that MLNs scale to large data sets.

Unfortunately, none of the current MLN implementations scale beyond

relatively small data sets (and even on many of these data sets, existing implementations routinely take hours to run). The first obvious reason is that these are *in-memory* implementations: when manipulating large intermediate data structures that overflow main memory, they either crash or thrash badly. Consequently, there is an emerging effort across several research groups to scale up MLNs. In this paper, we describe our system, Tuffy[1], that leverages an RDBMS to address the above scalability and performance problems.

There are two aspects of MLNs: learning and inference [103]. We focus on inference, since typically a model is learned once, and then an application may perform inference many times using the same model; hence inference is an on-line process, which must be fast. Moreover, MLN learning algorithms typically invoke inference as a subroutine repeatedly. Conceptually, inference[2] in MLNs has two phases: a *grounding phase*, which constructs a large, weighted SAT formula, and a *search phase*, which searches for a low cost (weight) assignment (called a *solution*) to the SAT formula from grounding (using WalkSAT [58], a local search procedure). Grounding is a non-trivial portion of the overall inference effort: on a classification benchmark (called RC) the state-of-the-art MLN inference engine, Alchemy[3], spends over 96% of its execution time in grounding. The state-of-the-art strategy for the grounding phase (and the one used by Alchemy) is a top-down procedure (similar to the proof strategy in Prolog). In contrast, we propose a bottom-up grounding strategy. Intuitively, bottom-up grounding allows Tuffy to fully exploit the RDBMS optimizer, and thereby significantly speed up the grounding phase of MLN inference. On an entity resolution task, Alchemy takes over 7 hours to complete grounding, while Tuffy's grounding finishes in less than 2

[1] `http://hazy.cs.wisc.edu/tuffy/`

[2] We focus on *maximum a posteriori inference* which is critical for many integration tasks.

[3] `http://alchemy.cs.washington.edu`

minutes.

But not all phases are well-optimized by the RDBMS: during the search phase, we found that the RDBMS implementation performed poorly. The underlying reason is a fundamental problem for pushing local search procedures into an RDBMS: search procedures often perform *inherently sequential, random* data accesses. Consequently, any RDBMS-based solution must execute a large number of disk accesses, each of which has a substantial overhead (due to the RDBMS) versus direct main-memory access. Not surprisingly, given the same amount of time, an in-memory solution can execute between three and five orders of magnitude more search steps than an approach that uses an RDBMS. Thus, to achieve competitive performance, we developed a novel hybrid architecture that supports local search procedures in main memory whenever possible. This is our second technical contribution.

Our third contribution is a simple partitioning technique that allows Tuffy to introduce parallelism and use less memory than state-of-the-art approaches. Surprisingly, this same technique often allows Tuffy to speed up the search phase exponentially. The underlying idea is simple: in many cases, a local search problem can be divided into multiple independent subproblems. For example, the formula that is output by the grounding phase may consist of multiple connected components. On such datasets, we derive a sufficient condition under which solving the subproblems independently results in exponentially faster search than running the larger global problem (Thm. 5.2). An application of our theorem shows that on an information extraction testbed, a system that is not aware of this phenomenon (such as Alchemy) must take at least 2^{200} more steps than Tuffy to reach a solution with the same quality. Empirically we found that, on some real-world datasets, solutions found by Tuffy within one minute have higher quality than those found by non-partitioning systems (such as Alchemy) even after running for days.

The exponential difference in running time for independent subproblems versus the larger global problem suggests that in some cases, further decomposing the search space may improve the overall runtime. To implement this idea for MLNs, we must address two difficult problems: (1) partitioning the formula from grounding (and so the search space) to minimize the number of formula that are split between partitions, and (2) augmenting the search algorithm to be aware of partitioning. We show that the first problem is NP-hard (even to approximate), and design a scalable heuristic partitioning algorithm. For the second problem, we apply a technique from non-linear optimization to leverage the insights gained from our characterization of the phenomenon described above. The effect of such partitioning is dramatic. As an example, on a classification benchmark (called RC), Tuffy (using 15MB of RAM) produces much better result quality in minutes than Alchemy (using 2.8GB of RAM) even after days of running. In fact, Tuffy is able to answer queries on a version of the RC dataset that is over two orders of magnitude larger. (We estimate that Alchemy would need 280GB+ of RAM to process it.)

Contributions, Validation, and Outline To summarize, we make the following contributions:

- In Section 5.2.1, we design a solution that pushes MLNs into RDBMSes. The key idea is to use bottom-up grounding that allows us to leverage the RDBMS optimizer; this idea improves the performance of the grounding phase by several orders of magnitude.
- In Section 5.2.2, we devise a novel hybrid architecture to support efficient grounding and in-memory inference. By itself, this architecture is far more scalable and, given the same amount of time, can perform orders of magnitude more search steps than prior art.

- In Section 5.2.3, we describe novel data partitioning techniques to decrease the memory usage and to increase parallelism (and so improve the scalability) of Tuffy's in-memory inference algorithms. Additionally, we show that for any MLN with an MRF that contains multiple components, partitioning could exponentially improve the expected (average case) search time.
- In Section 5.2.4, we generalize our partitioning results to arbitrary MLNs using our characterization of the partitioning phenomenon. These techniques result in our highest quality, most space-efficient solutions.

We present an extensive experimental study on a diverse set of MLN testbeds to demonstrate that our system Tuffy is able to get better result quality more quickly and work over larger datasets than the state-of-the-art approaches.

5.2 Tuffy Systems

In this section, we describe our technical contributions: a bottom-up grounding approach to fully leverage the RDBMS (Section 5.2.1); a hybrid main-memory RDBMS architecture to support efficient end-to-end inference (Section 5.2.2); and data partitioning which dramatically improves Tuffy's space and time efficiency (Section 5.2.3 and Section 5.2.4).

5.2.1 Grounding with a Bottom-up Approach

We describe how Tuffy performs grounding. In contrast to top-down approaches (similar to Prolog) that employ nested loops and that is used by prior MLN systems such as Alchemy, Tuffy takes a bottom-up approach (similar to Datalog) by expressing grounding as a sequence of SQL queries. Each SQL query is optimized by the RDBMS, which allows Tuffy to com-

plete the grounding process orders of magnitude more quickly than prior approaches.

For each predicate $P(\bar{A})$ in the input MLN, Tuffy creates a relation $R_P(\underline{aid}, \bar{A}, truth)$ where each row a_p represents an atom, aid is a globally unique identifier, $\bar{A}$ is the tuple of arguments of P, and $truth$ is a three-valued attribute that indicates if a_p is true or false (in the evidence), or not specified in the evidence. These tables form the input to grounding, and Tuffy constructs them using standard bulk-loading techniques.

In Tuffy, we produce an output table $C(\underline{cid}, lits, weight)$ where each row corresponds to a single ground clause. Here, cid is the id of a ground clause, $lits$ is an array that stores the atom id of each literal in this clause (and whether or not it is negated), and $weight$ is the weight of this clause. We first consider a formula without existential quantifiers. In this case, the formula F can be written as $F(\bar{x}) = l_1 \vee \cdots \vee l_N$ where $\bar{x}$ are all variables in F. Tuffy produces a SQL query Q for F that joins together the relations corresponding to the predicates in F to produce the atom ids of the ground clauses (and whether or not they are negated). The join conditions in Q enforce variable equality inside F, and incorporate the pruning strategies described in Section 2.3.3. For more details on the compilation procedure see Appendix A.1.1.

5.2.2 A Hybrid Architecture for Inference

Our initial prototype of Tuffy runs both grounding and search in the RDBMS. While the grounding phase described in the previous section has good performance and scalability, we found that performing search in an RDBMS is often a bottleneck. Thus, we design a hybrid architecture that allows efficient in-memory search (in Java) while retaining the performance benefits of RDBMS-based grounding. To see why in-memory search is critical, recall that WalkSAT works by selecting an unsatisfied clause C, selecting an atom in C, and "flipping" that atom to satisfy C. Thus, Walk-

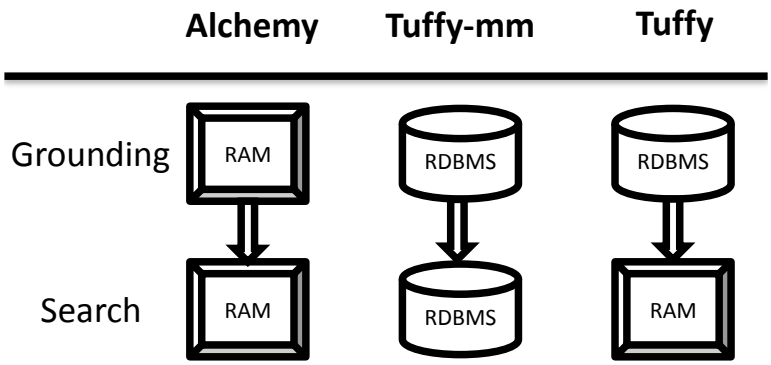


Figure 5.1: *Comparison of architectures*

SAT performs a large number of random accesses to the data representing ground clauses and atoms. Moreover, the data that is accessed in one iteration depends on the data that is accessed in the previous iteration. And so, this access pattern prevents both effective caching and parallelism, which causes a high overhead per data access. Thus, we implement a hybrid architecture where the RDBMS performs grounding and Tuffy is able to read the result of grounding from the RDBMS into memory and perform inference. If the grounding result is too large to fit in memory, Tuffy invokes an implementation of search directly inside the RDBMS (Appendix A.1.2). This approach is much less efficient than in-memory search, but it runs on datasets larger than main memory without crashing. Figure 5.1 illustrates the hybrid memory management approach of Tuffy. Alchemy is a representative of prior art MLN systems, which uses RAM for both grounding and search; Tuffy-mm is a version of Tuffy we developed that uses an RDBMS for all memory management; and Tuffy is the hybrid approach

While it is clear that this hybrid approach is at least as scalable as a direct memory implementation (such as Alchemy), there are in fact cases where Tuffy can run in-memory search whereas Alchemy would crash. The reason is that the space requirement of a purely in-memory

implementation is determined by the peak memory footprint throughout grounding *and* search, whereas Tuffy needs main memory only for search. For example, on a dataset called Relational Classification (RC), Alchemy allocated 2.8 GB of RAM only to produce 4.8 MB of ground clauses. On RC, Tuffy uses only 19 MB of RAM.

5.2.3 Partitioning to Improve Performance

In the following two sections, we study how to further improve Tuffy's space and time efficiency without sacrificing its scalability. The underlying idea is simple: we will try to partition the data. By splitting the problem into smaller pieces, we can reduce the memory footprint and introduce parallelism, which conceptually breaks the sequential nature of the search. These are expected benefits of partitioning. An unexpected benefit is an exponentially increase of the effective search speed, a point that we return to below.

First, observe that the logical forms of MLNs often result in an MRF with multiple disjoint components (see Appendix A.1.3). For example, on the RC dataset there are 489 components. Let G be an MRF with components $G_1, \cdots, G_k$; let I be a truth assignment to the atoms in G and I_i its projection over G_i. Then, it's clear that $\forall I$

$$\text{cost}^{G}_{\text{MLN}}(I) = \sum_{1 \leqslant i \leqslant k} \text{cost}^{G_i}_{\text{MLN}}(I_i).$$

Hence, instead of minimizing $\text{cost}^{G}_{\text{MLN}}(I)$ directly, it suffices to minimize each individual $\text{cost}^{G_i}_{\text{MLN}}(I_i)$. The benefit is that, even if G itself does not fit in memory, it is possible that each G_i does. As such, we can solve each G_i with in-memory search one by one, and finally merge the results together.[4]

[4] Alchemy exploits knowledge-based model construction (KBMC) [142] to find the minimal subgraph of the MRF that is needed for a given query. Alchemy, however, does not use the fact that the MRF output by KBMC may contain several components.

Component detection is done after the grounding phase and before the search phase, as follows. We maintain an in-memory union-find structure over the nodes, and scan the clause table while updating this union-find structure. The result is the set of connected components in the MRF. An immediate issue raised by partitioning is I/O efficiency.

Efficient Data Loading Once an MRF is split into components, loading in and running inference on each component sequentially one by one may incur many I/O operations, as there may be many partitions. For example, the MRF of the Information Extraction (IE) dataset contains thousands of 2-cliques and 3-cliques. One solution is to group the components into batches. The goal is to minimize the total number of batches (and thereby the I/O cost of loading), and the constraint is that each batch cannot exceed the memory budget. This is essentially the bin packing problem, and we implement the *First Fit Decreasing* algorithm [135]. Once the partitions are in memory, we can take advantage of parallelism. We use a round-robin scheduling policy.

Improving Search Speed using Partitioning Although processing each component individually produces solutions that are no worse than processing the whole graph at once, we give an example to illustrate that component-aware processing may result in exponentially faster speed of search.

Example 5.1. *Consider an MRF consisting of* N *identical connected components each containing two atoms* $\{X_i, Y_i\}$ *and three weighted clauses*

$$\{(X_i, 1), (Y_i, 1), (X_i \vee Y_i, -1)\},$$

where $i = 1 \ldots N$ *and the second component of each tuple is the weight. Based on how WalkSAT works, it's not hard to show that, if* $N = 1$*, starting from a random*

state, the expected hitting time[5] *of the optimal state, i.e.* $X_1 = Y_1 = \mathtt{True}$*, is no more than* 4*. Therefore, if we run WalkSAT on each component separately, the expected runtime of reaching the optimum is no more than* $4N$*. Now consider the case where we run WalkSAT on the whole MRF. Intuitively, reaching the optimal state requires "fixing" suboptimal components one by one. As the number of optimal components increases, however, it becomes more and more likely that one step of WalkSAT "breaks" an optimal component instead of fixing a suboptimal component. Such check and balance makes it very difficult for WalkSAT to reach the optimum. Indeed, Appendix A.1.5 shows that the expected hitting time is at least* 2^N *– an exponential gap!*

Let G be an MRF with components $G_1, \ldots, G_N$. *Component-aware WalkSAT* runs WalkSAT except that for each G_i, it keeps track of the lowest-cost state it has found so far on that G_i. In contrast, regular WalkSAT simply keeps the best overall solution it has seen so far. For $i = 1, \ldots, N$, let O_i be the set of optimal states of G_i, and S_i the set of non-optimal states of G_i that differ only by one bit from some $x^* \in O_i$; let $P_i(x \to y)$ be the transition probability of WalkSAT running on G_i, i.e., the probability that one step of WalkSAT would take G_i from x to y. Given x, a state of G_i, denote by $\nu_i(x)$ the number of violated clauses in G_i at state x; define

$$\alpha_i(x) = \sum_{y \in O_i} P_i(x \to y), \ \beta_i(x) = \sum_{y \in S_i} P_i(x \to y).$$

For any non-empty subset $H \subseteq \{1, \ldots, N\}$, define

$$r(H) = \frac{\min_{i \in H} \min_{x \in O_i} \nu_i(x)\beta_i(x)}{\max_{i \in H} \max_{x \in S_i} \nu_i(x)\alpha_i(x)}.$$

Theorem 5.2. *Let* H *be an arbitrary non-empty subset of* $\{1, \ldots, N\}$ *s.t.* $|H| \geqslant 2$ *and* $r = r(H) > 0$*. Then, in expectation, WalkSAT on* G *takes at least* $2^{|H|r/(2+r)}$

[5] The hitting time is a standard notion from Markov Chains [36], it is a random variable for the number of steps taken by WalkSAT to reach an optimum for the first time.

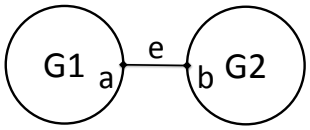


Figure 5.2: *A loosely connected graph for Example 5.3*

more steps to find an optimal solution than component-aware WalkSAT.

The proof is in Appendix A.1.5. In the worst case, there is only one component, or $r(H) = 0$ for every subset of components H (which happens only if there is a zero-cost solution), and partitioning would become pure overhead (but negligible in our experiments). On an information extraction (IE) benchmark dataset, there is some H with $|H| = 1196$ and $r(H) = 0.5$. Thus, the gap on this dataset is at least $2^{200} \approx 10^{60}$. This explains why Tuffy produces lower cost solutions in minutes than non-partition aware approaches such as Alchemy produce even after days.

5.2.4 Further Partitioning MRFs

Although our algorithms are more scalable than prior approaches, if the largest component does not fit in memory then we are forced to run the in-RDBMS version of inference, which is inefficient. Intuitively, if the graph is only weakly connected, then we should still be able to get the exponential speed up of partitioning. Consider the following example.

Example 5.3. *Consider an MRF consisting of two equally sized subgraphs G_1 and G_2, plus an edge $e = (a, b)$ between them (Figure 5.2). Suppose that the expected hitting time of WalkSAT on G_i is H_i. Since H_1 and H_2 are essentially independent, the hitting time of WalkSAT on G could be roughly $H_1 H_2$. On the other hand, consider the following scheme: enumerate all possible truth assignments to one of the boundary variables $\{a, b\}$, say a – of which there are two – and conditioning on each assignment, run WalkSAT on G_1 and G_2 independently. Clearly, the overall hitting time is no more than $2(H_1 + H_2)$, which is a huge improvement*

over $H_1 H_2$ *since* H_i *is usually a high-order polynomial or even exponential in the size of* G_i.

To capitalize on this idea, we need to address two challenges: 1) designing an efficient MRF partitioning algorithm; and 2) designing an effective partition-aware search algorithm. We address each of them in turn.

MRF Partitioning Intuitively, to maximally utilize the memory budget, we want to partition the MRF into roughly equal sizes; to minimize information loss, we want to minimize total weight of clauses that span over multiple partitions, i.e., the *cut size*. To capture this notion, we define a balanced bisection of a hypergraph $G = (V, E)$ as a partition of $V = V_1 \cup V_2$ such that $|V_1| = |V_2|$. The cost of a bisection (V_1, V_2) is $|\{e \in E | e \cap V_1 \neq \emptyset \text{ and } e \cap V_2 \neq \emptyset\}|$.

Theorem 5.4. *Consider the MLN* Γ *given by the single rule* $p(x), r(x, y) \rightarrow p(y)$ *where* r *is an evidence predicate. Then, the problem of finding a minimum-cost balanced bisection of the MRF that results from* Γ *is* NP*-hard in the size of the evidence (data).*

The proof (Appendix A.1.6) is by reduction to the graph minimum bisection problem [60], which is hard to approximate (unless $P = NP$, there is no PTAS). In fact, the problem we are facing (*multi-way hypergraph partitioning*) is more challenging than graph bisection, and has been extensively studied [56, 118]. And so, we design a simple, greedy partitioning algorithm: it assigns each clause to a bin in descending order by clause weight, subject to the constraint that no component in the resulting graph is larger than an input parameter β. We include pseudocode in Appendix A.1.7.

Partition-aware Search We need to refine the search procedure to be aware of partitions: the central challenge is that a clause in the cut may depend on atoms in two distinct partitions. Hence, there are dependencies

between the partitions. We exploit the idea in Example 5.3 to design the following partition-aware search scheme – which is an instance of the Gauss-Seidel method from nonlinear optimization [12, pg. 219]. Denote by $X_1, \ldots, X_k$ the states (i.e., truth assignments to the atoms) of the partitions. First initialize $X_i = x_i^0$ for $i = 1 \ldots k$. For $t = 1 \ldots T$, for $i = 1 \ldots k$, run WalkSAT on x_i^{t-1} conditioned on $\{x_j^t | 1 \leqslant j < i\} \cup \{x_j^{t-1} | i < j \leqslant k\}$ to obtain x_i^t. Finally, return $\{x_i^T | 1 \leqslant i \leqslant k\}$.

Tradeoffs Although fine-grained partitioning improves per-partition search speed (Theorem 3.1) and space efficiency, it also increases cut sizes – especially for dense graphs – which would in turn slow down the Gauss-Seidel inference scheme. Thus, there is an interesting tradeoff of partitioning granularity. In Section B.8, we describe a basic heuristic that combines Theorem 3.1 and the Gauss-Seidel scheme.

5.3 Experiments

In this section, we validate first that our system Tuffy is orders of magnitude more scalable and efficient than prior approaches. We then validate that each of our techniques contributes to the goal.

Experimental Setup We select Alchemy, the currently most widely used MLN system, as our comparison point. Alchemy and Tuffy are implemented in C++ and Java, respectively. The RDBMS used by Tuffy is PostgreSQL 8.4. Unless specified otherwise, all experiments are run on an Intel Core2 at 2.4GHz with 4 GB of RAM running Red Hat Enterprise Linux 5. For fair comparison, in all experiments Tuffy runs a single thread unless otherwise noted.

Datasets We run Alchemy and Tuffy on four datasets; three of them (including their MLNs) are taken directly from the Alchemy website[6]: *Link Prediction* (**LP**), given an administrative database of a CS department, the goal is to predict student-adviser relationships; *Information Extraction* (**IE**), given a set of Citeseer citations, the goal is to extract from them structured records; and *Entity Resolution* (**ER**), which is to deduplicate citation records based on word similarity. These tasks have been extensively used in prior work. The last task, *Relational Classification* (**RC**), performs classification on the Cora dataset [77]; **RC** contains all the rules in Figure 2.6. Table 6.5 contains statistics about the data.

	LP	**IE**	**RC**	**ER**
#relations	22	18	4	10
#rules	94	1K	15	3.8K
#entities	302	2.6K	51K	510
#evidence tuples	731	0.25M	0.43M	676
#query atoms	4.6K	0.34M	10K	16K
#components	1	5341	489	1

Table 5.1: *Dataset statistics*

5.3.1 High-level Performance

We empirically demonstrate that Tuffy with all the techniques we have described has faster grounding, higher search speed, lower memory usage, and in some cases produces much better solutions than a competitor main memory approach, Alchemy. Recall that the name of the game is to produce low-cost solutions quickly. With this in mind, we run Tuffy and Alchemy on each dataset for 7500 seconds, and track the cost of the best solution found up to any moment; on datasets that have multiple components, namely IE and RC, we apply the partitioning strategy in

[6] `http://alchemy.cs.washington.edu`

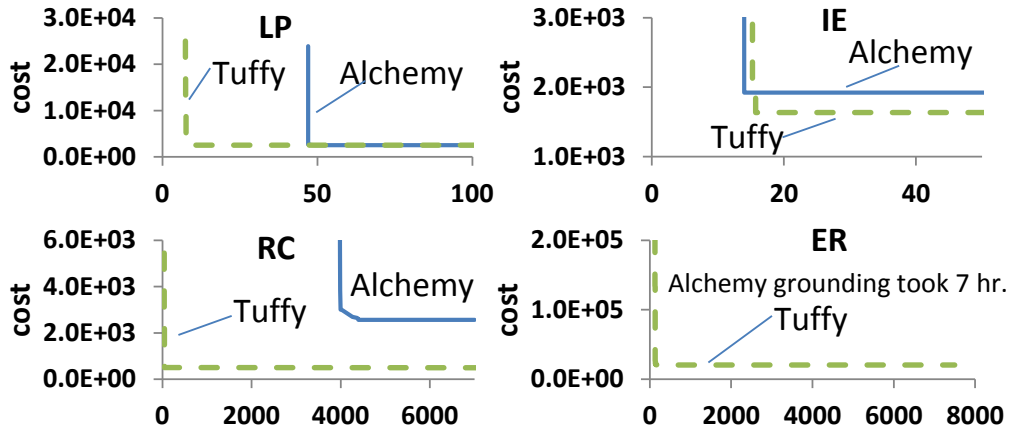


Figure 5.3: *Time-cost plots of* Alchemy *vs.* Tuffy*; the* x *axes are time (sec).*

Section 5.2.3 on Tuffy. As shown in Figure 5.3, Tuffy often reaches a best solution within orders of magnitude less time than Alchemy; secondly, the result quality of Tuffy is at least as good as – sometimes substantially better (e.g., on IE and RC) than – Alchemy. Here, we have zoomed the time axes into interesting areas. Since "solution cost" is undefined during grounding, each curve begins only when grounding is completed[7]. We analyze the experiment results in more detail in the following sections.

5.3.2 Effect of Bottom-up Grounding

We validate that the RDBMS-based grounding approach in Tuffy allows us to complete the grounding process orders of magnitude more efficiently than Alchemy. To make this point, we run Tuffy and Alchemy on the four datasets, and show their grounding time in Table 5.2. We can see that Tuffy outperforms Alchemy by orders of magnitude at run time in the grounding phase (a factor of 225 on the ER dataset). To understand the

[7]The L-shaped curves indicate that search converges very quickly compared to grounding time.

	LP	IE	RC	ER
Alchemy	48	13	3,913	23,891
Tuffy	6	13	40	106

Table 5.2: *Grounding time (sec)*

differences, we dug deeper with a lesion study (i.e., disabling one aspect of a system at a time), and found that sort join and hash join algorithms (along with predicate pushdown) are the key components of the RDBMS that speeds up the grounding process of Tuffy (Appendix A.2.2). Tuffy obviates the need for Alchemy to reimplement the optimization techniques in an RDBMS.

5.3.3 Effect of Hybrid Architecture

We validate two technical claims: (1) the hybrid memory management strategy of Tuffy (even without our partitioning optimizations) has comparable search rates to existing main memory implementations (and much faster than RDBMS-based implementation) and (2) Tuffy maintains a much smaller memory footprint (again without partitioning). Thus, we compare three approaches: (1) Tuffy without the partitioning optimizations, called Tuffy-p (read: Tuffy minus p), (2) a version of Tuffy (also without partitioning) that implements RDBMS-based WalkSAT (detailed in Appendix A.1.2), Tuffy-mm, and (3) Alchemy.

Figure 5.4 illustrates the time-cost plots on LP and RC of all three approaches. We see from RC that Tuffy-p is able to ground much more quickly than Alchemy (40 sec compared to 3913 sec). Additionally, we see that, compared to Tuffy-mm, Tuffy-p's in-memory search is orders of magnitude faster at getting to their best reported solution (both approaches finish grounding at the same time, and so start search at the same time). To understand why, we measure the *flipping rate*, which is the number of steps performed by WalkSAT per second. As shown in Table 5.3, the

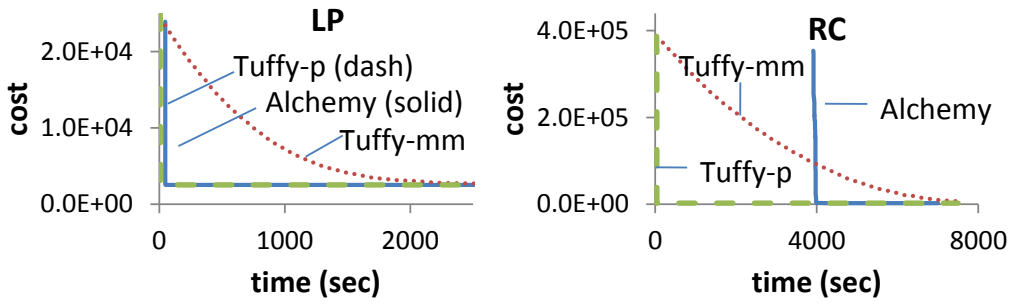


Figure 5.4: *Time-cost plots of* Alchemy *vs.* Tuffy*-p (i.e.,* Tuffy *without partitioning) vs.* Tuffy*-mm (i.e.,* Tuffy *with RDBMS-based search)*

	LP	IE	RC	ER
Alchemy	0.20M	1M	1.9K	0.9K
Tuffy-mm	0.9	13	0.9	0.03
Tuffy-p	0.11M	0.39M	0.17M	7.9K

Table 5.3: *Flipping rates (#flips/sec)*

	LP	IE	RC	ER
clause table	5.2 MB	0.6 MB	4.8 MB	164 MB
Alchemy RAM	411 MB	206 MB	2.8 GB	3.5 GB
Tuffy-p RAM	9 MB	8 MB	19 MB	184 MB

Table 5.4: *Space efficiency of* Alchemy *vs.* Tuffy*-p (without partitioning)*

reason is that Tuffy-mm has a dramatically lower flipping rate. We discuss the performance bound of any RDBMS-based search implementation in Appendix A.2.1.

To validate our second claim, that Tuffy-p has a smaller memory footprint, we see in Table 5.4, that on all datasets, the memory footprint of Tuffy is no more than 5% of Alchemy. Drilling down, the reason is that the intermediate state size of Alchemy's grounding process may be larger than the size of grounding results. For example, on the RC dataset, Alchemy

allocated 2.8 GB of RAM only to produce 4.8 MB of ground clauses. While Alchemy has to hold everything in memory, Tuffy only needs to load the grounding result from the RDBMS at the end of grounding. It follows that, given the same resources, there are MLNs that Tuffy can handle efficiently while Alchemy would crash. Indeed, on a dataset called "ER+" which is twice as large as ER, Alchemy exhausts all 4GB of RAM and crashes soon after launching, whereas Tuffy runs normally with peak RAM usage of roughly 2GB.

From these experiments, we conclude that the hybrid architecture is crucial to Tuffy's overall efficiency.

5.3.4 Effect of Partitioning

In this section, we validate that, when there are multiple components in the data, partitioning not only improves Tuffy's space efficiency, but – due to Theorem 5.2 – may actually enable Tuffy to find substantially higher quality results. We compare Tuffy's performance (with partitioning enabled) against Tuffy-p: a version of Tuffy with partitioning disabled.

We run the search phase on each of the four datasets using three approaches: Alchemy, Tuffy-p, and Tuffy (with partitioning). Tuffy-p and Alchemy run WalkSAT on the whole MRF for 10^7 steps. Tuffy runs WalkSAT on each component in the MRF independently, each component G_i receiving $10^7|G_i|/|G|$ steps, where $|G_i|$ and $|G|$ are the numbers of atoms in this component and the MRF, respectively. This is weighted round-robin scheduling.

As shown in Table 5.5, when there are multiple components in the MRF, partitioning allows Tuffy to use less memory than Tuffy-p. (The IE dataset is too small to yield notable differences). We see that Tuffy's component-aware inference can produce significantly better results than Tuffy-p. We then extend the run time of all systems. As shown in Figure 5.5, there continues to be a gap between Tuffy's component-aware search approach

	LP	IE	RC	ER
#components	1	5341	489	1
Tuffy-p RAM	9MB	8MB	19MB	184MB
Tuffy RAM	9MB	8MB	15MB	184MB
Tuffy-p cost	2534	1933	1943	18717
Tuffy cost	2534	1635	1281	18717

Table 5.5: *Performance of* Tuffy *vs.* Tuffy-*p (i.e.,* Tuffy *without partitioning)*

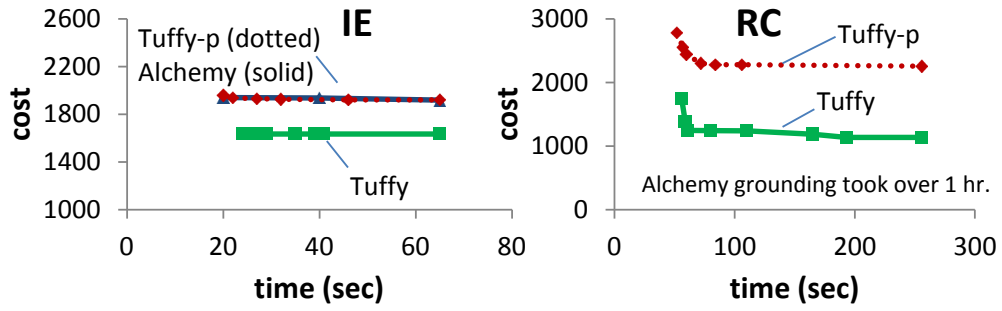


Figure 5.5: *Time-cost plots of* Tuffy *vs* Tuffy-*p (i.e.,* Tuffy *without partitioning)*

and the original WalkSAT running on the whole MRF. This gap is predicted by our theoretical analysis in Section 5.2.3. Thus, we have verified that partitioning makes Tuffy substantially more efficient in terms of both space and search speed.

We also validate that Tuffy's loading and parallelism makes a substantial difference: without our batch loading technique, Tuffy takes 448s to perform 10^6 search steps per component on RC, while 117s to perform the same operation with batch loading. With the addition of 8 threads (on 8 cores), we further reduce the runtime to 28s. Additional loading and parallelism experiments in Appendix A.2.3 support our claim that our loading algorithm and partitioning algorithm contribute to improving processing speed.

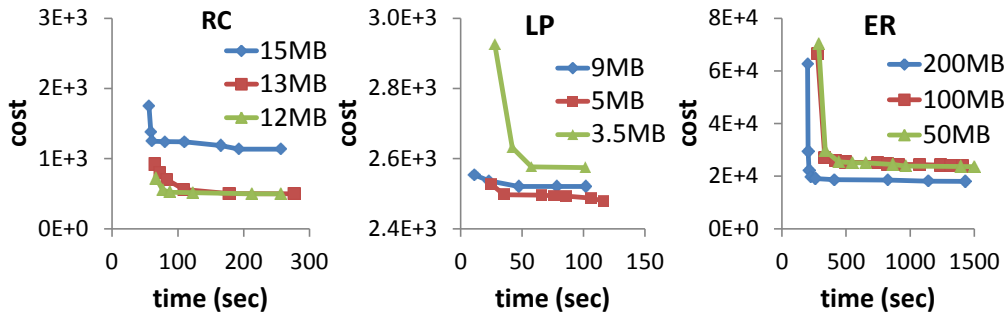


Figure 5.6: *Time-cost plots of* Tuffy *with different memory budgets*

5.3.5 Effect of Further Partitioning

To validate our claim that splitting MRF components can further improve both space efficiency and sometimes also search quality (Section 5.2.4), we run Tuffy on RC, ER, and LP with different memory budgets – which are fed to the partitioning algorithm as the bound of partition size. On each dataset, we give Tuffy three memory budgets, with the largest one corresponding to the case when no components are split. Figure 5.6 shows the experiment results. On RC, we see another improvement of the result quality (cf. Figure 5.5). Similar to Example 5.3, we believe the reason to be graph sparsity: "13MB" cuts only about 420 out of the total 10K clauses. In contrast, while MRF partitioning lowers RAM usage considerably on ER, it also leads to slower convergence – which correlates with poor partitioning quality: the MRF of ER is quite dense and even 2-way partitioning ("100MB") would cut over 1.4M out of the total 2M clauses. The dataset LP illustrates the interesting tradeoff where a coarse partition is beneficial whereas finer grained partitions would be detrimental. We discuss this tradeoff in Appendix A.1.8.

5.4 Summary

Motivated by a large set of data-rich applications, we study how to push MLN inference inside an RDBMS. We find that the grounding phase of MLN inference performs many relational operations and that these operations are a substantial bottleneck in state-of-the-art MLN implementations such as Alchemy. Using an RDBMS, Tuffy not only achieves scalability, but also speeds up the grounding phase by orders of magnitude. We then develop a hybrid solution with RDBMS-based grounding and in-memory search. To improve the space and time efficiency of Tuffy, we study a partitioning approach that allows for in-memory search even when the dataset does not fit in memory. We showed that further partitioning allows Tuffy to produce higher quality results in a shorter amount of time.

6 Scaling Markov Logic via Task Decomposition

The central observation of this chapter is as follows: A KBC task usually involves routine subtasks such as classification and coreference resolution. Moreover, such subtasks may correspond to a subset of rules in an MLN program. Thus, instead of running a generic MLN inference algorithm on the entire program (as done by state-of-the-art MLN inference algorithms), we can partition the rule set into subtasks and invoke specialized algorithms for corresponding subtasks. Most of the work in this chapter appears in Niu, Zhang, Ré, and Shavlik [88, 90].

6.1 Motivations, Challenges, and Contributions

Markov logic [103] is a knowledge-representation language that uses weighted first-order logic to specify graphical models [63]. It has been applied to a wide range of applications, including many in information extraction and data mining [4, 97, 110, 126, 149]. The resulting graphical models can be huge (hundreds of millions of nodes or more in our applications), and so a key technical challenge is the scalability and performance of Markov logic inference.

Semantically, a Markov logic program, or *Markov logic network (*MLN*)*, specifies a graphical model called a *Markov random field* (MRF). Thus, one approach to improving the scalability of MLNs is to apply inference techniques from the graphical-model literature. One such technique is *dual decomposition* [123, 143] from the mathematical programming literature that has recently been applied to MRF inference [54, 64, 66, 136]. In addition to increased scalability, dual decomposition also offers the possibility of dual certificates[1] that can bound the distance of a solution

[1]Namely an (exact or approximate) lower (resp. upper) bound to a minimization

from optimality, e.g., for *maximum a posteriori (MAP)* inference.

There are two steps in applying dual decomposition: (1) decompose the inference problem into multiple parts, and (2) iteratively combine solutions from individual parts. The intuition is that the individual parts will be more tractable than the whole problem – so much so that the improved performance of individual parts will compensate for the overhead of many iterations of repeated inference. Following the literature [64, 66], we first implement MRF-level decomposition and compare it with state-of-the-art MLN inference algorithms. On simpler MLN-generated MRFs, MRF-level decomposition can achieve competitive performance compared to monolithic inference (i.e., running generic MLN inference algorithm without decomposition). On more complex MLN-generated MRFs, the performance and quality of both monolithic inference and MRF decomposition approaches may be suboptimal.

Our key observation is that MRF-level decomposition strategies in step (1) ignore valuable structural hints that often occur in MLN programs. For example, a large Markov logic program may have several "subroutines" that each perform a standard, well-studied task such as coreference resolution or labeling, e.g., with a *conditional random field* (CRF) [70]. In contrast to traditional approaches that either use a generic MLN inference algorithm or decompose into semantically meaningless parts like trees, our idea is to exploit this information so that we may use existing, specialized algorithms for such individual subtasks. For example, we may choose to solve a labeling task with a CRF and so use the Viterbi algorithm [70]. Importantly, even if we use different algorithms for each part, dual decomposition preserves the joint-inference property – i.e., different parts are not solved in isolation. The hope is to perform higher-quality, more scalable inference than previous monolithic approaches.

To illustrate this idea, we describe several such correspondences be-

(resp. maximization) problem.

	MRF-level	**Program-level**
Input	MRF component	MLN rule set
Parts	Trees	Subprograms (Tasks)
Inference	Max-product	Task-specific
Multipliers	Special factors	Singleton rules

Table 6.1: *Comparing MRF-level and program-level dual decomposition approaches to MLN inference.*

tween MLN structure and specialized inference algorithms, including logistic regression and linear-chain conditional random fields. A user may declare (or an algorithm may detect) which of these specialized algorithms to use on an individual part of the MLN program. We call our prototype system that integrates these specialized algorithms Felix.

Experimentally, we validate that our (MRF-level and program-level) dual-decomposition approaches have superior performance than prior MLN inference approaches on several data-mining tasks. Our main results are that (1) on simple MLNs taken from the literature, MRF-level decomposition outperforms monolithic inference, and Felix has competitive efficiency and quality compared to monolithic inference and MRF-level decomposition; and (2) on more complex MLNs (also taken from the literature), Felix achieves substantially higher efficiency and quality than monolithic inference and MRF-level decomposition.

6.2 Dual Decomposition for MLNs

The dual decomposition technique (Section 2.3.4) leaves open the question of *how* to decompose a function f. The two approaches that we implement for dual decomposition work at different levels of abstraction: at the MRF level or at the MLN-program level. Still, the two methods are similar: both pass messages in a master-slave scheme, and produce a MAP solution after inference in a similar way. As a result, we are able to implement both

approaches on top of the Tuffy [87] system. Table 6.1 provides a high-level comparison between these two approaches. We describe each step of the process in turn: decomposition (Section 6.2.1), master-slave message passing (Section 6.2.2), and producing the final solution (Section 6.2.3).

6.2.1 Decomposition

In decomposition, we partition the input structure and set up auxiliary structures to support message passing. For MRF-level decomposition, we partition an MRF into multiple trees that are linked via auxiliary singleton factors; for program-level decomposition, we partition an MLN into subprograms that are linked via auxiliary singleton rules.

MRF-level Decomposition The input to the decomposition algorithm is an MRF which is the result of grounding an input MLN. The MRF is represented as a *factor graph* (i.e., a bipartite graph between ground-tuple nodes and ground-formula factors).[2] Following Komodakis et al. [65, 66] and Wainwright et al. [136], we decompose the MRF into a collection of trees (smaller factor graphs with disjoint factors) that cover this factor graph, i.e., each node in the factor graph is present in one or more trees. Nodes that are in more than one tree may take conflicting values; resolving these conflicts is the heart of the message-passing procedure in the next phase.

We decompose each component of this factor graph independently. The goal is to partition the factors (including their incident edges) into several groups so that each group forms a tree. To do this, we run a greedy algorithm that begins by choosing a ground-formula factor at random. We create a tree with this factor, and then iteratively pick a neighboring factor that shares at least one node with some factor in this tree. We check if adding this neighboring factor to the tree would introduce cycles. If not, we

[2] This representation allows for non-pairwise MRFs.

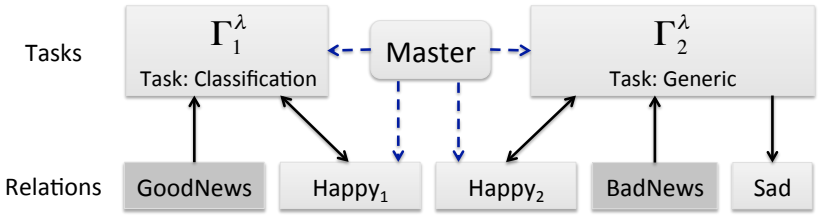


Figure 6.1: *A program-level decomposition for Example 6.1. Shaded boxes are evidence relations. Solid arrows indicate data flow; dash arrows are control.*

add the neighboring factor to the tree and proceed to the next neighboring factor (in a breadth-first manner). Once all neighboring factors have been exhausted, we remove this tree from the factor graph, and then repeat the above process for the next tree.

Some nodes may be shared by multiple trees. To allow messages to be passed in the next step, we create a special singleton factor for each copy of a shared node. The weight of this factor represents a Lagrange multiplier and may change between iterations.

Program-level Decomposition In contrast, Felix performs decomposition at the program-level: the input is the (first-order) rule set of an MLN program, and Felix partitions it into multiple tasks each of which can be solved with different algorithms. One design decision that we make in the program-level approach is that entire relations are shared or not – as opposed to individual ground tuples. This choice allows Felix to use a relational database management system (RDBMS) for all data movement, which can be formulated as SQL queries; in turn this allows us to deal with low-level issues like memory management. We illustrate the idea with an example.

Example 6.1. *Consider a simple MLN which we call* Γ*:*

$$
\begin{array}{ll}
1 & \texttt{GoodNews}(p) \Rightarrow \texttt{Happy}(p) \quad \phi_1 \\
1 & \texttt{BadNews}(p) \Rightarrow \texttt{Sad}(p) \quad \phi_2 \\
5 & \texttt{Happy}(p) \Leftrightarrow \neg\texttt{Sad}(p) \quad \phi_3
\end{array}
$$

where GoodNews *and* BadNews *are evidence and the other two relations are queries. Consider the decomposition* $\Gamma_1 = \{\phi_1\}$ *and* $\Gamma_2 = \{\phi_2, \phi_3\}$. Γ_1 *and* Γ_2 *share the relation* Happy*; so we create two copies of this relation:* $\texttt{Happy}_1$ *and* $\texttt{Happy}_2$*, one for each subprogram. We introduce Lagrange multipliers* λ_p*, one for each possible ground tuple* $\texttt{Happy}(p)$*. We thereby obtain a new program* Γ^λ*:*

$$
\begin{array}{ll}
1 & \texttt{GoodNews}(p) \Rightarrow \texttt{Happy}_1(p) \quad \phi_1' \\
\lambda_p & \texttt{Happy}_1(p) \quad \varphi_1 \\
1 & \texttt{BadNews}(p) \Rightarrow \texttt{Sad}(p) \quad \phi_2 \\
5 & \texttt{Happy}_2(p) \Leftrightarrow \neg\texttt{Sad}(p) \quad \phi_3' \\
-\lambda_p & \texttt{Happy}_2(p) \quad \varphi_2
\end{array}
$$

where each φ_i *represents a set of singleton rules, one for each value of* p *(i.e., for each specific person in a given testbed). This program contains two subprograms,* $\Gamma_1^\lambda = \{\phi_1', \varphi_1\}$ *and* $\Gamma_2^\lambda = \{\phi_2, \phi_3', \varphi_2\}$*, that can be solved independently with any inference algorithm.*

As illustrated in Figure 6.1, the output of our decomposition method is a bipartite graph between a set of subprograms and a set relations. In a program-level approach, Felix attaches an inference algorithm to each subprogram; we call this pair of algorithm and subprogram a *task*. We discuss how to select decompositions and assign algorithms in Section 6.3.

6.2.2 Message Passing

We apply the master-slave message passing scheme [65, 66] for both MRF-level and program-level decomposition. The master-slave approach alternates between two steps: (1) perform inference on each part independently to obtain each part's predictions on shared variables, and (2) a process

called the *Master* examines the (possibly conflicting) predictions and sends messages in the form of Lagrange multipliers to each task. Below we describe this process using the notation from program-level decomposition. Essentially, the same algorithm applies to MRF-level decomposition; we describe the minor differences at the end of the section.

The Master chooses the values of the Lagrange multipliers via an optimization process [65, 66]. In particular, we can view the Master as optimizing $\max_\lambda g(\lambda)$ using an iterative solver, in our case projected subgradient ascent [143, p. 174]. Specifically, let p be a tuple of a query relation r; in the given decomposition, p may be shared between k subprograms and so there are k copies of p – call them $p_1, \ldots, p_k$. For $i = 1, \ldots, k$, let $p_i^t \in \{0, 1\}$ denote the value of p predicted by task i at step t and λ_i^t denote the corresponding Lagrange multiplier. At step t, the Master updates λ_i^t by comparing the predicted value of p_i by task i to the average value for p output by all inference tasks. This leads to the following update rule:

$$\lambda_i^{t+1} := \lambda_i^t + \alpha_t \left(p_i^t - \frac{|\{j : p_j = 1\}|}{k} \right),$$

where α_t is the step size for this update. Following standard practice in numerical optimization [5], in this paper's experiments, we use the diminishing step size rule $\alpha_t = 1/t$ for both types of decomposition. The subgradient ascent procedure stops either when all copies have reached an agreement or when Felix has run a pre-specified maximum number of iterations.

MRF-level Approach Message passing for MRF-level decomposition follows the same master-slave scheme as above. A difference is that to perform inference on each tree, we can perform *exact* inference (to predict each p_i) by running the max-product algorithm [67]. A key benefit of this approach is that it allows tractable computation of a dual certificate. We

have implemented dual certificates, but postpone them to the technical-report version as they are not essential to our contribution.

6.2.3 Producing the Final Solution

Inference is a continuous process, so some copies of variables may not have converged at the end of a run. The last step is to choose a final solution based on solutions from the decomposed parts.

MRF-level Decomposition To obtain a solution to the original MLN from individual solutions on each tree, we use the heuristic proposed by Komogorov et al. [64] (and used in Komodakis et al. [66]) that sequentially fixes shared-variable values based on max-product messages in individual trees.

Program-level Decomposition If a shared relation is not subject to any hard rules, Felix takes majority votes from the predictions of related tasks. (If all copies of this relation have converged, the votes would be unanimous.) To ensure that hard rules in the input MLN program are not violated in the final output, we insist that for any query relation r, all hard rules involving r (if any) be assigned to a single task, and that the final value of r be taken from this task.[3] This guarantees that the final output is a possible world for Γ (provided that the hard rules are satisfiable).

6.3 Specialized Tasks

With program-level decomposition, we can use different algorithms for different subprograms. Felix uses specialized algorithms to handle tasks, which can be more efficient than generic MLN inference. As an existence

[3] This policy might result in cascaded subtasks. More sophisticated policies are an interesting future direction.

Task	Implementation
Classification	Logistic Regression [13]
Segmentation	Conditional Random Fields [70]
Coreference	Correlation Clustering [2, 8]

Table 6.2: *Example specialized tasks and their implementations in* Felix.

proof, we describe several tasks that are common in text processing (see Table 6.2). We leave more comprehensive taxonomies of such tasks as future work.

6.3.1 Individual Tasks

We describe each task, how it arises in an MLN, a specialized algorithm, and an informal argument describing why being aware of the special structure may outperform MRF-level decomposition or monolithic approaches.

Classification Classification is a fundamental statistical problem and ubiquitous in applications. Classification arises in Markov logic as a query predicate $R(x, y)$ with hard rules of the form

$$R(x, y_1) \wedge y_1 \neq y_2 => \neg R(x, y_2),$$

which mandates that each object (represented by a possible value of x) can only be assigned at most one label (represented by a possible value of y). For example, the rules F_6 and F_7 in Figure 2.6 form a classification task that determines whether each `affil` tuple (considered as an object with Boolean labels) holds. If the only query relation in a subprogram Γ_i is R and R is mentioned at most once in each rule in Γ_i (except the rule above), then Γ_i is essentially a *logistic regression* (LR) classification model. The inference problem for LR is trivial given model parameters (here rule weights) and feature values (here ground formulae).

On the other hand, suppose there are N objects and K labels. Then

it would require $N\binom{K}{2}$ factors to represent the above rule in an MRF. For tasks such as entity linking (e.g., mapping textual mentions to Wikipedia entities), the value of K could be in the millions.

Segmentation Conditional random fields (CRFs) [70] have been widely applied to text applications. In particular, linear-chain CRFs are a popular way for performing word-sequence segmentation and labeling, and can be solved efficiently with dynamic programming algorithms. A linear-chain CRF model consists of unigram features and bigram features. In Markov logic, unigram features have the same form as LR rules, while bigram features are soft rules of the form

$$L(x_1, x_2) => R(x_1, y_1) \wedge R(x_2, y_2),$$

where L is an evidence relation containing correlated object pairs (e.g., consecutive tokens in a sentence). If the correlation structure L represents chains over the objects (i.e., x values), then we can solve a subproblem with unigram and bigram rules like these with the Viterbi algorithm. In contrast, the MRF representation (with Boolean variables) of this model is not a chain, and so cannot be solved with Viterbi.

Coreference Another common task is coreference resolution (coref), e.g., given a set of N strings (say phrases in a document) we want to decide which strings represent the same real-world entity. This arises in MLNs as a query relation R that is subject to hard rules encoding an equivalent relation, including the reflexivity, symmetry, and transitivity properties. The input to a coreference task is a single relation $B(o1, o2, wgt)$ where $wgt = \beta_{o1,o2} \in \mathbb{R}$ indicates how likely the objects o1, o2 are coreferent (with 0 being neutral). The output of a coreference task is an instance of relation $R(o1, o2)$ that indicates which pairs of objects are coreferent. Assuming that $\beta_{o1,o2} = 0$ if $(o1, o2)$ is not covered by the relation B, then

Properties	Symbol	Example
Reflexive	REF	$p(x,y) \implies p(x,x)$
Symmetric	SYM	$p(x,y) \implies p(y,x)$
Transitive	TRN	$p(x,y), p(y,z) \implies p(x,z)$
Key	KEY	$p(x,y), p(x,z) \implies y = z$
Not Recursive	NoREC	Can be defined w/o recursion.
Tree Recursive	TrREC	Tree-structured MRF

Table 6.3: *Properties assigned to predicates by the* Felix *compiler.*

each valid R incurs a cost (called *disagreement cost*)

$$\text{cost}_{\text{coref}}(R) = \sum_{\substack{o1,o2:(o1,o2)\notin R \\ \text{and } \beta_{o1,o2}>0}} |\beta_{o1,o2}| + \sum_{\substack{o1,o2:(o1,o2)\in R \\ \text{and } \beta_{o1,o2}<0}} |\beta_{o1,o2}|.$$

In Figure 2.6, F_1 through F_5 can be mapped to a coreference task for the relation `pCoref`. F_1 through F_3 encode the reflexivity, symmetry, and transitivity properties of `pCoref`, and the ground formulae of F_4 and F_5 specify the weighted-edge relation B. The goal is to find a relation R that achieves the minimum cost.

Coreference is a well-studied problem [35], and there are approximate inference techniques for coreference resolution [2, 8, 119]. Felix implements both *correlation clustering algorithms* [2] and Singh et al.'s sampling algorithm [119]. In contrast, explicitly representing a coreference task in an MRF may be inefficient: a direct implementation of transitivity requires N^3 factors.

6.3.2 Task Detection

In our prototype, we allow a user to manually decompose an MLN program and specify a specific inference algorithm to be run on each subprogram. Ideally Felix should be able to automatically recognize MLN

Task	Required Properties
Classification	KEY, NoREC
Segmentation	KEY, TrREC
Coreference	REF, SYM, TRN
Generic MLN Inference	none

Table 6.4: *Tasks and their required properties.*

subprograms that could be processed as specialized tasks. In this section we describe a best-effort compiler that is able to automatically detect the presence of classification, segmentation, and coref tasks.[4] If a subprogram does not match any specialized algorithm, we run a generic MLN inference algorithm (WalkSAT [58]).[5]

To automatically decompose an MLN program Γ into tasks, Felix uses a two-step approach. Felix's first step is to annotate each query predicate p with a set of *properties*. An example property is whether or not p is symmetric. Table 6.3 lists of the set of properties that Felix attempts to discover with their definitions. Once the properties are found, Felix uses Table 6.4 to list all possible options for a predicate. When there are multiple options, the current prototype of Felix simply chooses the first task to appear in the following order: (Coref, Classification, Segmentation, Generic). This order intuitively favors more specific tasks. To compile an MLN into tasks, Felix greedily applies the above procedure to split a subset of rules into a task, and then iterates until all rules have been consumed.

The most technically difficult part of the compiler is determining the properties of the predicates. There are two types of properties that Felix looks for: (1) schema-like properties of any possible worlds that satisfy Γ and (2) graphical structures of correlations between tuples.

[4] As we will see in Table 6.6, our automatic matching algorithm is able to find five out of the six decompositions in the experiments.

[5] Alternatively we could also run MRF-level dual decomposition on this subprogram.

Schema Properties Although the set of properties in Table 6.3 is motivated by considerations from statistical inference, the first four properties depend *only on the hard rules in* Γ. To detect these schema-like properties, FELIX uses a set of sound (but not complete) rules that are described by simple patterns. For example, we can conclude that a predicate R is transitive if program contains syntactically the rule $R(x,y), R(y,z) => R(x,z)$ with weight ∞.

Graphical Structure The second type of properties that FELIX considers characterize the graphical structure of the *ground database* (in turn, this structure describes the correlations that must be accounted for in the inference process). The ground database is a function of both soft and hard rules in the input program, and so we consider both types of rules here. FELIX's compiler attempts to deduce a special case of recursion that is motivated by (tree-structured) conditional random fields that we call TrREC. Suppose that there is a single recursive rule that contains p in both the body and the head is of the form:

$$p(x,y), T(y,z) => p(x,z) \tag{6.1}$$

where the first attribute of T is a key and the transitive closure of T is a partial order. In the ground database, p will be "tree-structured". MAP and marginal inference for such rules are in P-time [137]. FELIX has a regular expression to deduce this property.

6.4 Experiments

Our main hypotheses are that (1) MRF-level decomposition can outperform monolithic inference; and (2) on smaller datasets, FELIX achieves similar performance to MRF-level decomposition and monolithic inference, while on larger datasets it achieves higher performance than MRF-level

decomposition and monolithic inference. We validate them on several testbeds.

We also validate that specialized algorithms indeed have higher efficiency and quality than generic MLN inference algorithms or MRF decomposition; therefore being able to integrate specialized algorithms is a key reason for FELIX's high performance.

Datasets and MLNs We use four publicly available MLN testbeds from ALCHEMY's website.[6] In addition, we create an MLN program for named-entity recognition based on skip-chain CRFs[129], and we describe an MLN program that was developed for TAC-KBP, a knowledge-base population challenge.[7] Table 6.5 shows some statistics of these datasets.

We describe the MLN testbeds from ALCHEMY's website: (1) **IE**, where one performs segmentation on Cora citations using unigram and bigram features (see the "Isolated" program [97]). (2) **IERJ**, where one performs joint segmentation and entity resolution on Cora citations (see the "Jnt-Seg-ER" program [97]). (3) **ER**, where one performs entity resolution on four predicates with pairwise signals as well as inter-predicate correlation rules (see the "B+N+C+T" program [120]). (4) **WebKB**, where one predicts (non-exclusive) categories of webpages from academic departments based on content words and links between pages [76].

The last two are (5) **NER**, where one performs named-entity recognition on a dataset with 10K tokens using the skip-chain CRF model [129] (encoded in three MLN rules), and (6) **KBP**, which is an implementation of the TAC-KBP (knowledge-base population) challenge using an MLN that performs entity linking (mapping textual mentions to Wikipedia entities), slot filling (mapping co-occurring mentions to a set of possible relationships), entity-level knowledge-base population, and fact verification from an existing partial knowledge base. Using this MLN, the results for TAC-

[6] `http://alchemy.cs.washington.edu/`
[7] `http://nlp.cs.qc.cuny.edu/kbp/2010/`

	Relations	Rules	Evidence	MRF	DB
IE	17	34	150K	137K	11MB
IERJ	18	357	150K	564K	44MB
ER	10	29	1.5K	19M	1.4GB
WebKB	3	57	282K	520K	41MB
NER	2	4	10K	1.7M	134MB
KBP	7	6	4.3M	20M	1.6GB
KBP+	7	6	240M	64B	5.1TB

Table 6.5: *Dataset sizes. The columns are the number of relations in the input MLN, the number of MLN rules, the number of evidence tuples, and the number of MRF factors after grounding with* Tuffy*, and the size of in-database representation of the MRF.*

KBP are comparable to the state of the art [53] – we achieved a F1 score of 0.80 on entity linking (human performance is about 0.90) and 0.31 on slot filling.

Among those datasets, IE, WebKB, and NER are simpler programs as they only have unigram and sparse bigram rules and classification constraints with very few labels; IERJ, ER, and KBP are more complex programs as they involve transitivity rules and classification constraints with many labels. To test the scalability of Felix, we also run the KBP program on a 1.8M-doc TAC-KBP corpus ("**KBP+**").

Experimental Setup We run Tuffy [87] and Alchemy as state-of-the-art monolithic MLN inference systems. We implement MRF-level decomposition and program-level decomposition (i.e., Felix) approaches on top of the open-source Tuffy system.[8] As Tuffy has similar or superior performance to Alchemy on each dataset, here we use Tuffy as a representative for state-of-the-art MLN inference and report Alchemy's performance in the technical-report version of this paper. We use the following labels

[8] `http://hazy.cs.wisc.edu/tuffy`

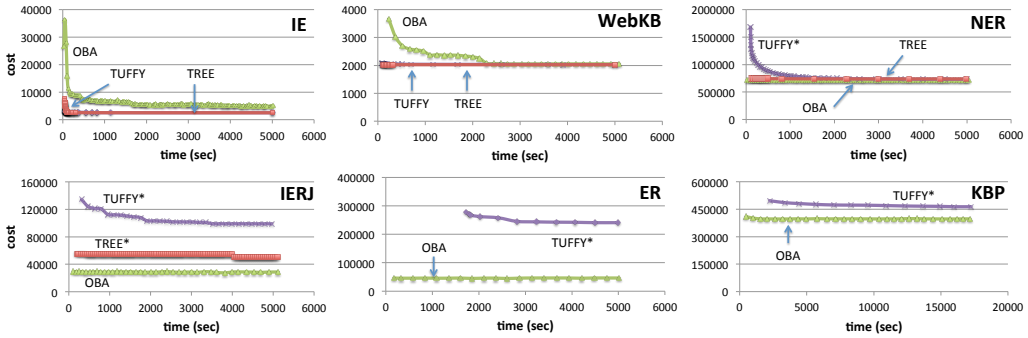


Figure 6.2: *High-level performance results of various approaches to MLN inference. For each dataset and each system, we plot a time-cost curve. TREE ran out of memory on ER and KBP. Labels ending with an asterisk indicate that some points in the corresponding curves correspond to infeasible solutions (i.e., solutions with infinite cost), and the curves were obtained by "softening" hard rules to have weight 100.*

for these three approaches: (1) **TUFFY**, in which Tuffy performs RDBMS-based grounding (i.e., MLN-to-MRF transformation) and uses WalkSAT as its inference algorithm; (2) **TREE**, in which we replace Tuffy's WalkSAT algorithm with tree-based MRF-level dual decomposition as described in Section 6.2; and (3) **OBA**, in which we implement program-level dual decomposition as described Sections 6.2 and 6.3; we use Singh et al.'s algorithm [119] for Coref. Table 6.6 lists Felix's decomposition scheme for each dataset.

All three approaches are implemented in Java and use PostgreSQL 9.0.4 as the underlying database system. Unless specified otherwise, all experiments are run on a RHEL 6.1 server with four 2.00GHz Intel Xeon CPUs (40 total physical cores plus hyperthreading) and 256 GB of RAM. Although all three approaches can be parallelized on most datasets, we use single-thread runtime when plotting graphs. Recall that we use the step size rule $\alpha_t = \frac{1}{t}$. It is known that alternative step size rules may result in faster convergence, and we report a study of these step size rules in the

	LR	CRF	Coref	WalkSAT
IE	1	0	0	1
IERJ	1	0	1	1
ER	0	0	4	1
WebKB	0	0	0	2
NER	0	1	0	1
KBP	2	0	0	1

Table 6.6: *Number of tasks of each type in the decompositions used by* Felix. *Our automatic matching algorithm finds all decompositions except for* ***WebKB****.*

technical-report version of this paper.

6.4.1 Overall Efficiency and Quality

We validate that TREE can outperform TUFFY and that Felix in turn outperforms both TREE and TUFFY on complex programs. To support these claims, we compare the efficiency and quality of all three approaches on the datasets listed above. We run each system on each dataset for 5000 seconds (except for the largest dataset KBP, for which we run 5 hours), and plot the MLN cost against runtime.

From Figure 6.2 we see that, on the three simpler programs (i.e., IE, WebKB, and NER), all three approaches were able to converge to about the same result quality. Although TREE and TUFFY have similar performance on IE and WebKB, on NER the TREE approach obtains a low-cost solution within two minutes whereas TUFFY takes more than one hour to reach comparable quality. Felix has slower convergence behavior than TUFFY and TREE on IE and WebKB, but it does converge to a similar solution. We note that alternate step size rules may significantly improve Felix's performance on these datasets.

On the more complex programs (i.e., IERJ, ER, and KBP), Felix achieves dramatically better performance compared to TUFFY and TREE: while

TUFFY and TREE fail to find a feasible solution (i.e., a solution with finite cost) after 5000 seconds on each of these datasets, Felix converges to a feasible solution within minutes on each dataset. There are complex structures in these MLNs; e.g., transitivity for entity resolution and uniqueness constraints for entity linking in KBP. TUFFY and TREE were not able to find feasible solutions that satisfy such complex constraints, and the corresponding curves were obtained by replacing hard rules with a "softened" version with weight 100. Still, we see that the results from both TUFFY and TREE are substantially worse than Felix. From the above comparison, we conclude that overall the Felix approach is able to achieve significantly higher efficiency and quality than TUFFY and TREE approaches.

Scalability To test scalability, we also run Felix on the large KBP+ dataset with a parallel RDBMS (from Greenplum Inc.). This MLN converges within a few iterations; an iteration takes about five hours in Felix.

6.4.2 Specialized Tasks

We next validate that the ability to integrate specialized tasks into MLN inference is key to Felix's higher efficiency and quality. To do this, we demonstrate that Felix's specialized algorithms outperform generic MLN inference algorithms in both quality and efficiency when solving specialized tasks. To evaluate this claim, we run Felix, Tuffy, TREE, and Alchemy on three MLN programs encoding the following tasks: logistic regression-based classification (LR), linear-chain CRF-based classification (CRF), and correlation clustering (CC). To measure application quality (F1 scores), we select some datasets with ground truth: we use a subset of the Cora dataset[9] for CC, and a subset of the CoNLL 2000 chunking dataset[10] for LR and CRF. As shown in Table 6.7, while it always takes less than a minute

[9] `http://alchemy.cs.washington.edu/data/cora`

[10] `http://www.cnts.ua.ac.be/conll2000/chunking/`

Task	System	Initial	Final	Cost	F1
LR	Felix	21 s	21 s	2.4e4	0.79
	Tuffy	58 s	59 s	2.4e4	0.79
	TREE	58 s	196 s	2.4e4	0.79
	Alchemy	3140 s	3152 s	3.4e4	0.14
CRF	Felix	35 s	35 s	4.6e5	0.90
	Tuffy	148 s	186 s	∞ (6.4e5)	0.14 (0.14)
	TREE	150 s	911 s	4.7e5	0.16
	Alchemy	740 s	760 s	1.4e6	0.10
CC	Felix	11 s	11 s	1.8e4	0.35
	Tuffy	977 s	1730 s	∞ (2.0e4)	0.33 (0.32)
	TREE	982 s	11433 s	∞ (3.3e4)	0.16 (0.16)
	Alchemy	2622 s	2640 s	∞ (4.6e5)	0.54 (0.49)

Table 6.7: *Performance and quality comparison on individual tasks. "Initial" (resp. "Final") is the time when a system produced the first (resp. converged) result. "F1" is the F1 score of the final output. For system-task pairs with infinite cost, we also "soften" hard rules with a weight 100, and report corresponding cost and F1 in parentheses. Each cost/F1 value is an average over five runs.*

for Felix on each task, the other approaches take much longer. Moreover, Felix has the best inference quality (i.e., cost) and application quality (i.e., F1).[11]

6.5 Summary

We study how to apply dual decomposition to Markov logic inference. We find that MRF-level decomposition empirically outperforms traditional, monolithic approaches to MLN inference on some programs. However, MRF-level decomposition ignores valuable structural hints in Markov logic programs. Thus, we propose an alternative decomposition strategy that partitions an MLN program into high-level tasks (e.g., classification) that

[11]The only exception is Alchemy's F1 on CC, an indication that the CC program is suboptimal for the application.

can be solved with specialized algorithms. On several datasets, we empirically show that our program-level decomposition approach outperforms both monolithic inference and MRF-level decomposition approaches to MLN inference.

Our future work is in several directions. First, we plan to investigate how to algorithmically use dual certificates in program-level dual decomposition, e.g., to support early stopping. The current prototype of Felix uses classical schemes for the iterative steps of DD, and so the convergence rate could be further improved by using recent techniques [42, 54]. To support broader applications, we plan to extend Felix with new tasks. An interesting challenge is to develop an architecture that allows one to quickly and easily add in new inference techniques. Another interesting topic is how to automatically perform decomposition and select inference algorithms for an MLN based on program structure. To improve the efficiency of non-specialized tasks, we could apply existing MLN inference techniques, e.g., lifted inference. Lastly, our prototype implementation of Felix executes each task from scratch at each iteration; incremental or warm-start inference could improve the overall efficiency.

7 Conclusion and Future Work

7.1 Conclusion

This dissertation demonstrates that the distant supervision technique for statistical learning and the Markov logic framework for statistical inference are indeed effective approaches to web-scale KBC. The technical contributions include the Elementary architecture to knowledge-base construction, a systematic study of distant supervision, and two novel techniques for scaling Markov logic inference. Using Elementary, we built a system called DeepDive that reads hundreds of millions of web pages to enhance Wikipedia with facts of tens of relations. We have also deployed Elementary to two additional domains: AncientText, where we help English professors study 140K books from the 18th century, and GeoDeepDive, where we help geoscientists organize information in 20K journal papers in geology.

7.2 Future Work

Beyond the current prototype of Elementary, there are several interesting future research directions. We discuss them below.

The Crowd Crowdsourcing did not display a strong impact on the quality of distant supervision in our previous study [148]. In our experiments, we only asked the crowd to verify the examples obtained via distant supervision. However, there are many other types of input that one could take from the crowd. It is plausible that certain types of input from the crowd could have much larger impacts than we observed. For example, one may solicit new examples or even new features (e.g., linguistic patterns) via crowdsourcing. It is interesting to explore possibilities like these.

Inference Efficiency There are two types of computational redundancies in the inference systems of Elementary. First, the message passing scheme in Felix currently is an iterative process that invokes individual inference algorithms to execute from scratch at each iteration. Second, developing KBC systems is also an iterative process, and so the underlying models of a KBC system (say, an MLN) would evolve over time; currently Elementary would need to perform all inference tasks from scratch upon such changes. Ideally we would like to address these inefficiencies, say using some form of incremental inference.

Statistical Modeling Languages Because a Markov logic program consists of logical formulae, the graphical model compiled from an MLN only consists of CNF or DNF factors. This limitation is a key reason for the representational and computational inefficiency of monolithic MLN inference systems like Alchemy and Tuffy. For example, if we represent a liner-chain CRF model with more than two labels in Markov logic, the graphical model compiled from the Markov logic program would no longer be form a chain. The reason is that the MLN-based graphical model uses multiple Boolean variables and hard constraints to essentially emulate one multinomial variable in the CRF model. Although Felix is able to opportunistically address such issues by "reverse-engineering" such inefficient representations in MLNs, a cleaner solution would be to avoid such representations altogether. Furthermore, certain types of correlations (e.g., COUNT) that can be useful for a statistical KBC system cannot be easily expressed in an MLN. Addressing the above issues requires careful thoughts on a new language for specifying graphical models.

Feature Engineering Markov logic (or any flexible framework for statistical inference) provides the *ability* to integrate diverse input signals (or statistical features), but not all features have the same impact on the quality of KBC. From our experience of developing DeepDive and GeoDeepDive,

we have found that the process of debugging and tuning features tends to be iterative and have humans in the loop. This feature engineering process is often tedious and painstaking. An interesting direction is to provide tools to assist developers with feature engineering; e.g., automatic parameter tuning and smart diagnosis of an Elementary pipeline.

Bibliography

[1] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94. ACM, 2000.

[2] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *JACM*, 55:23, 2008.

[3] D. Allen and A. Darwiche. New advances in inference by recursive conditioning. In *Proceeding of the Conference on Uncertainty in Artificial Intelligence*, 2003.

[4] D. Andrzejewski, L. Livermore, X. Zhu, M. Craven, and B. Recht. A framework for incorporating general domain knowledge into latent Dirichlet allocation using first-order logic. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, 2011.

[5] K. Anstreicher and L. Wolsey. Two âŁšÃ„Ãºwell-knownâŁšÃ„Ã¹ properties of subgradient optimization. *Mathematical Programming*, 120(1):213–220, 2009.

[6] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *Proceedings of the International Conference on Data Engineering*, 2008.

[7] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348. ACM, 2003.

[8] A. Arasu, C. Ré, and D. Suciu. Large-scale deduplication with constraints using Dedupalog. In *Proceedings of the International Conference on Data Engineering*, 2009.

[9] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1):89–113, 2004.

[10] O. Benjelloun, A. Sarma, A. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *Proceedings of International Conference on Very Large Data Bases*, 17:243–264, 2008.

[11] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[12] D. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1989.

[13] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, 2004.

[14] R. D. S. Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 1319–1325, 2005.

[15] S. Brin. Extracting patterns and relations from the world wide web. In *Proceedings of the International Conference on World Wide Web*, pages 172–183. Springer, 1999.

[16] A. Calì, G. Gottlob, and A. Pieris. Query answering under expressive entity-relationship schemata. *Conceptual Modeling–ER 2010*, 1:347–361, 2010.

[17] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka Jr, and T. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Conference on Artificial Intelligence*, pages 1306–1313, 2010.

[18] X. Chai, B. Vuong, A. Doan, and J. Naughton. Efficiently incorporating user feedback into information extraction and integration

programs. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pages 87–100. ACM, 2009.

[19] F. Chen, X. Feng, R. Christopher, and M. Wang. Optimizing statistical information extraction programs over evolving text. In *Proceedings of the International Conference on Data Engineering*, volume 5, page 6, 2012.

[20] P. Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1:9–36, 1976.

[21] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, and S. Vaithyanathan. SystemT: An algebraic approach to declarative information extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2010.

[22] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the 1998 National Conference on Artificial Intelligence*, 1998.

[23] M. Craven and J. Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86, 1999.

[24] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *Proceedings of International Conference on Very Large Data Bases*, 2004.

[25] R. de Salvo Braz, E. Amir, and D. Roth. MPE and partial inversion in lifted probabilistic variable elimination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1123–1130, 2006.

[26] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceeding of the USENIX Symposium on Operating Systems Design and Implementation*, pages 137–150, 2004.

[27] E. Demaine, D. Emanuel, A. Fiat, and N. Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.

[28] P. DeRose, W. Shen, F. Chen, Y. Lee, D. Burdick, A. Doan, and R. Ramakrishnan. DBLife: A community information management platform for the database research community. In *Proceeding of the Conference on Innovative Data Systems Research*, 2007.

[29] A. Deshpande and S. Madden. MauveDB: Supporting model-based user views in database systems. *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, 1:73–84, 2006.

[30] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, 2009.

[31] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. Entity disambiguation for knowledge base population. In *ACL*, pages 277–285, 2010.

[32] J. Duchi, D. Tarlow, G. Elidan, and D. Koller. Using combinatorial optimization within max-product belief propagation. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2007.

[33] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of the 13th international conference on World Wide Web*, pages 100–110. ACM, 2004.

[34] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Center. Open information extraction: The second generation. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[35] I. Fellegi and A. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 1299:51, 1969.

[36] W. Feller. *An introduction to probability theory and its applications. Vol. I*. New York: John Wiley\ & Sons, 1950.

[37] X. Feng, A. Kumar, B. Recht, and C. Ré. Towards a unified architecture for in-RDBMS analytics. In *Proceedings of the SIGMOD International Conference on Management of Data*, 2012.

[38] B. R. Feng Niu, C. Ré, and S. J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, 2011.

[39] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. Murdock, E. Nyberg, J. Prager, et al. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.

[40] J. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 363–370, 2005.

[41] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, 1999.

[42] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. *Advances in Neural Information Processing Systems*, 21:1, 2007.

[43] M. Gormley, A. Gerber, M. Harper, and M. Dredze. Non-expert correction of automatically generated relation annotations. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 204–207, 2010.

[44] M. Greenwood and M. Stevenson. Improving semi-supervised acquisition of relation extraction patterns. In *Proceedings of the Workshop on Information Extraction beyond the Document*, pages 29–35. Association for Computational Linguistics, 2006.

[45] R. Gupta, A. Diwan, and S. Sarawagi. Efficient inference with cardinality-based clique potentials. In *Proceedings of the International Conference on Machine Learning*, 2007.

[46] M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics-Volume 2*, pages 539–545, 1992.

[47] J. Hilbe. *Logistic Regression Models*. CRC Press, 2009.

[48] J. Hoffart, M. A. Yosef, I. Bordino, H. FâˆšÂºrstenau, M. Pin, M. Spaniol, B. Taneva, S. Thater, G. Weikum, and G. Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792, 2011.

[49] R. Hoffmann, S. Amershi, K. Patel, F. Wu, J. Fogarty, and D. Weld. Amplifying community content creation with mixed initiative information extraction. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1849–1858. ACM, 2009.

[50] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. Weld. Knowledge-based weak supervision for information extraction of

overlapping relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 541–550, 2011.

[51] R. Hoffmann, C. Zhang, and D. Weld. Learning 5000 relational extractors. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 286–295, 2010.

[52] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. Jermaine, and P. Haas. MCDB: A monte carlo approach to managing uncertain data. In *Proceedings of the SIGMOD International Conference on Management of Data*, 2008.

[53] H. Ji, R. Grishman, H. Dang, K. Griffitt, and J. Ellis. Overview of the TAC 2010 knowledge base population track. In *Text Analysis Conference*, 2010.

[54] V. Jojic, S. Gould, and D. Koller. Accelerated dual decomposition for map inference. In *Proceedings of the International Conference on Machine Learning*, 2010.

[55] B. Kanagal and A. Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *Proceedings of the International Conference on Data Engineering*, 2008.

[56] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Applications in VLSI domain. *IEEE Transactions on VLSI Systems*, 7:69–79, 2002.

[57] G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum. The YAGO-NAGA approach to knowledge discovery. *SIGMOD Record*, 37(4):41–47, 2008.

[58] H. Kautz, B. Selman, and Y. Jiang. A general stochastic approach to solving problems with hard and soft constraints. *The Satisfiability Problem: Theory and Applications*, 17:1, 1997.

[59] K. Kersting, B. Ahmadi, and S. Natarajan. Counting belief propagation. In *Proceeding of the Conference on Uncertainty in Artificial Intelligence*, 2009.

[60] S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *Proceeding of the IEEE Annual Symposium on Foundations of Computer Science*, 2004.

[61] T. Khot, S. Natarajan, K. Kersting, and J. Shavlik. Learning markov logic networks via functional gradient boosting. In *IEEE 11th International Conference on Data Mining (ICDM)*, pages 320–329, 2011.

[62] S. Kok and P. Domingos. Learning the structure of markov logic networks. In *Proceedings of the 22nd international conference on Machine learning*, pages 441–448. ACM, 2005.

[63] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[64] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[65] N. Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2985–2992, 2009.

[66] N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *Proceeding of the IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[67] F. Kschischang, B. Frey, and H. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[68] N. Kushmerick. *Wrapper induction for information extraction*. PhD thesis, University of Washington, 1997.

[69] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1):15–68, 2000.

[70] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.

[71] N. Lao, T. Mitchell, and W. Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539, 2011.

[72] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[73] D. Lin and P. Pantel. DIRT-discovery of inference rules from text. *Knowledge Discovery and Data Mining*, 1:323–328, 2001.

[74] D. Lin and P. Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[75] B. Liu, L. Chiticariu, V. Chu, H. Jagadish, and F. Reiss. Automatic rule refinement for information extraction. In *Proceedings of International Conference on Very Large Data Bases*, 2010.

[76] D. Lowd and P. Domingos. Efficient weight learning for Markov logic networks. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 200–211, 2007.

[77] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000.

[78] A. McCallum, K. Schultz, and S. Singh. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2009.

[79] E. Michelakis, R. Krishnamurthy, P. Haas, and S. Vaithyanathan. Uncertainty management in rule-based information extraction systems. In *Proceedings of the SIGMOD International Conference on Management of Data*, 2009.

[80] L. Mihalkova and R. Mooney. Bottom-up learning of Markov logic network structure. In *Proceedings of the International Conference on Machine Learning*, 2007.

[81] L. Mihalkova and M. Richardson. Speeding up inference in statistical relational learning by clustering similar query literals. In *Proceeding of the International Workshop on Inductive Logic Programming*, 2010.

[82] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1003–1011, 2009.

[83] R. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 328–334, 1999.

[84] N. Nakashole, M. Theobald, and G. Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the Web Search and Data Mining*, 2011.

[85] T. Nguyen and A. Moschitti. End-to-end relation extraction using distant supervision from external semantic repositories. In *Proceeding of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 277–282, 2011.

[86] T. Nguyen and A. Moschitti. Joint distant and direct supervision for relation extraction. In *Proceeding of the International Joint Conference on Natural Language Processing*, pages 732–740, 2011.

[87] F. Niu, C. Ré, A. Doan, and J. Shavlik. Tuffy: Scaling up statistical inference in Markov logic networks using an RDBMS. In *Proceedings of International Conference on Very Large Data Bases*, 2011.

[88] F. Niu, C. Zhang, C. Ré, and J. Shavlik. Felix: Scaling Inference for Markov Logic with an Operator-based Approach. *ArXiv e-prints*, 2011.

[89] F. Niu, C. Zhang, C. Ré, and J. Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *Second Int.l Workshop on Searching and Integrating New Web Data Sources*, 2012.

[90] F. Niu, C. Zhang, C. Ré, and J. Shavlik. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *International Journal On Semantic Web and Information Systems Special Issue on Web-Scale Knowledge Extraction*, 2012.

[91] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135, 2007.

[92] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings*

of the Annual Meeting of the Association for Computational Linguistics, pages 113–120. Association for Computational Linguistics, 2006.

[93] T. Park and A. Van Gelder. Partitioning methods for satisfiability testing on large formulas. *Information and Computation*, 62:179–184, 2000.

[94] J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1988.

[95] D. Poole. First-order probabilistic inference. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 985–991, 2003.

[96] H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2006.

[97] H. Poon and P. Domingos. Joint inference in information extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2007.

[98] H. Poon and P. Domingos. Joint unsupervised coreference resolution with Markov Logic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008.

[99] H. Poon and P. Domingos. Unsupervised semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–10. Association for Computational Linguistics, 2009.

[100] H. Poon, P. Domingos, and M. Sumner. A general method for reducing the complexity of relational inference and its application to MCMC. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2008.

[101] C. Ré, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *Proceedings of the International Conference on Data Engineering*, 2007.

[102] C. Ré, J. Letchner, M. Balazinska, and D. Suciu. Event queries on correlated probabilistic streams. In *Proceedings of the SIGMOD International Conference on Management of Data*, 2008.

[103] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[104] S. Riedel. Improving the accuracy and efficiency of MAP inference for Markov logic. In *Proceeding of the Conference on Uncertainty in Artificial Intelligence*, 2008.

[105] S. Riedel and I. Meza-Ruiz. Collective semantic role labeling with Markov logic. In *Proceedings of the Conference on Computational Natural Language Learning*, 2008.

[106] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*, pages 148–163, 2010.

[107] E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 811–811, 1993.

[108] A. Rush, D. Sontag, M. Collins, and T. Jaakkola. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–11, 2010.

[109] A. Sahuguet and F. Azavant. Building intelligent web applications using lightweight wrappers. *Data & Knowledge Engineering*, 36(3):283–316, 2001.

[110] S. Satpal, S. Bhadra, S. Sellamanickam, R. Rastogi, and P. Sen. Web information extraction using markov logic networks. In *KDD*, pages 1406–1414, 2011.

[111] P. Sen, A. Deshpande, and L. Getoor. Bisimulation-based approximate lifted inference. In *Proceeding of the Conference on Uncertainty in Artificial Intelligence*, pages 496–505, 2009.

[112] P. Sen, A. Deshpande, and L. Getoor. PrDB: Managing and exploiting rich correlations in probabilistic databases. *Proceedings of International Conference on Very Large Data Bases*, 18:1065–1090, 2009.

[113] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2012.

[114] B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, 2008.

[115] J. Shavlik and S. Natarajan. Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, 2009.

[116] W. Shen, A. Doan, J. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *Proceedings of International Conference on Very Large Data Bases*, 2007.

[117] V. Sheng, F. Provost, and P. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622, 2008.

[118] H. Simon and S. Teng. How good is recursive bisection? *SIAM Journal on Scientific Computing*, 18:1436–1445, 1997.

[119] S. Singh, A. Subramanya, F. Pereira, and A. McCallum. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Proceeding of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.

[120] P. Singla and P. Domingos. Entity resolution with Markov logic. In *Proceedings of the International Conference on Data Mining*, 2006.

[121] P. Singla and P. Domingos. Memory-efficient inference in relational domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2006.

[122] P. Singla and P. Domingos. Lifted first-order belief propagation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2008.

[123] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 1:1–11, 2010.

[124] M. Stevenson and M. Greenwood. Learning information extraction patterns using wordnet. In *Proceedings of the Third International WordNet Conference*, pages 95–102, 2006.

[125] F. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *Proceedings of the International Conference on World Wide Web*, pages 697–706. ACM, 2007.

[126] F. Suchanek, M. Sozio, and G. Weikum. SOFIE: A self-organizing framework for information extraction. In *Proceedings of the International Conference on World Wide Web*, 2009.

[127] M. Surdeanu and C. Manning. Ensemble models for dependency parsing: cheap and good? In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 649–652, 2010.

[128] M. Surdeanu, D. McClosky, J. Tibshirani, J. Bauer, A. Chang, V. Spitkovsky, and C. Manning. A simple distant supervision approach for the TAC-KBP slot filling task. In *Proceedings of Text Analysis Conference 2010 Workshop*, 2010.

[129] C. Sutton and A. McCallum. Collective segmentation and labeling of distant entities in information extraction. Technical Report 04-49, University of Massachusetts, 2004.

[130] C. Sutton and A. McCallum. *An introduction to conditional random fields for relational learning*. Introduction to statistical relational learning. MIT Press, 2006.

[131] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceeding of the Conference on Uncertainty in Artificial Intelligence*, 2002.

[132] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: The Condor experience. *Concurrency and Computation: Practice and Experience*, 17(2-4):323–356, 2005.

[133] M. Theobald, M. Sozio, F. Suchanek, and N. Nakashole. URDF: Efficient Reasoning in Uncertain RDF Knowledge Bases with Soft and Hard Rules. Technical report, MPI, 2010.

[134] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58:267–288, 1996.

[135] V. Vazirani. *Approximation algorithms*. Springer Verlag, 2001.

[136] M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.

[137] M. Wainwright and M. Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers, 2008.

[138] D. Wang, E. Michelakis, M. Garofalakis, and J. Hellerstein. Bayesstore: Managing large, uncertain data repositories with probabilistic graphical models. *Proceedings of the VLDB Endowment*, pages 340–351, 2008.

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

[140] G. Weikum and M. Theobald. From information to knowledge: Harvesting entities and relationships from web sources. In *Proceedings of the ACM Symposium on Principles of Database Systems*, 2010.

[141] D. Weld, R. Hoffmann, and F. Wu. Using Wikipedia to bootstrap open information extraction. *SIGMOD Record*, 37:62–68, 2009.

[142] M. Wellman, J. Breese, and R. Goldman. From knowledge bases to decision models. *The Knowledge Engineering Review*, 7:35–53, 1992.

[143] L. Wolsey. *Integer Programming*. Wiley, 1998.

[144] F. Wu and D. Weld. Autonomously semantifying wikipedia. In *ACM Conference on Information and Knowledge Management*, pages 41–50, 2007.

[145] F. Wu and D. Weld. Automatically refining the Wikipedia infobox ontology. In *Proceeding of the 17th international conference on World Wide Web*, pages 635–644. ACM, 2008.

[146] F. Wu and D. Weld. Open information extraction using Wikipedia. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 118–127, 2010.

[147] L. Yao, S. Riedel, and A. McCallum. Collective cross-document relation extraction without labelled data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023, 2010.

[148] C. Zhang, F. Niu, C. Ré, and J. Shavlik. Big data versus the crowd: Looking for relationships in all the right places. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2012.

[149] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J. Wen. Statsnowball: A statistical approach to extracting entity relationships. In *Proceedings of the International Conference on World Wide Web*, 2009.

A TUFFY

This chapter contains supplemental material for Chapter 5.

A.1 Additional Systems Details

A.1.1 A Compilation Algorithm for Grounding

Algorithm 11 is a basic algorithm of expressing the grounding process of an MLN formula in SQL. To support existential quantifiers, we used PostgreSQL's array aggregate feature.

Algorithm 11 MLN Grounding in SQL

Input: an MLN formula $\phi = \vee_{i=1}^{k} l_i$ where each l_i is a literal supported by predicate table $r(l_i)$
Output: a SQL query Q that grounds ϕ

1: FROM clause of Q includes '$r(l_i)$ t_i' for each literal l_i
2: SELECT clause of Q contains 't_i.aid' for each literal l_i
3: For each positive (resp. negative) literal l_i, there is a WHERE predicate 't_i.truth $\neq$ true' (resp. 't_i.truth $\neq$ false')
4: For each variable x in ϕ, there is a WHERE predicate that equates the corresponding columns of t_i's with l_i containing x
5: For each constant argument of l_i, there is an equal-constant WHERE predicate for table t_i
6: Form a conjunction with the above WHERE predicates

A.1.2 Implementing WalkSAT in RDBMS

WalkSAT is a stochastic local search algorithm; its random access patterns pose considerable challenges to the design of TUFFY. More specifically, the following operations are difficult to implement efficiently with on-disk data: 1) uniformly sample an unsatisfied clause; 2) random access (read/write) to per-atom or per-clause data structures; and 3) traverse

clauses involving a given atom. Atoms are cached as in-memory arrays, while the per-clause data structures are read-only. Each step of WalkSAT involves a scan over the clauses and many random accesses to the atoms.

Although our design process iterated over numerous combinations of various design choices, we were still unable to reduce the gap as reported in Section 5.3.2. For example, compared to clause table scans, one might suspect that indexing could improve search speed by reading less data at each step. However, we actually found that the cost of maintaining indices often outweighs the benefit provided by indexing. Moreover, we found it very difficult to get around RDBMS overhead such as PostgreSQL's mandatory MVCC.

A.1.3 MLNs Causing MRF Fragmentation

MLN rules usually model the interaction of relationships and attributes of some underlying entities. As such, one can define entity-based transitive closures, which directly corresponds to components in the MRF. Since in real world data the interactions are usually sparse, one can expect to see multiple components in the MRF. A concrete example is the paper classification running example, where the primary entities are papers, and the interactions are defined by citations and common authors. Indeed, our RC dataset yields hundreds of components in the MRF (see Table 5.5).

A.1.4 Data Loading as Bin Packing

The data loading problem as described in Section 5.2.3 is essentially the bin packing problem. Formally, given as input a set of items $I = \{i_1, \ldots, i_N\}$ with weights $w_1, \ldots, w_N$ and a size constraint M, the bin packing problem is to produce a partition $I = B_1 \cup \cdots \cup B_k$ such that $\sum_{i \in B_j} w_i \leqslant M$ for $j = 1, \ldots, k$ for the smallest possible k.

Theorem A.1 (From [135]). *Let* **OPT** *denote the number of bins in an optimal solution. For all* $\varepsilon > 0$*, there does not exist an algorithm that runs in polynomial time and produces* $(1.5 - \varepsilon)\mathbf{OPT}$ *bins for any input unless* $P = NP$*.*

TUFFY implements an approximation algorithm that uses at most $1.7\mathbf{OPT} + 2$ bins. It is an efficient algorithm called *First Fit Decreasing* [135]. It sorts the components (or their ids) by decreasing size, then it greedily puts as many items as possible in each batch (without violating the space budget).

A.1.5 Theorem 5.2

Proof of Theorem 5.2. We follow the notations of the theorem. Without loss of generality and for ease of notation, suppose $H = \{1, \ldots, N\}$. Denote by Ω the state space of G. Let $Q_k \subseteq \Omega$ be the set of states of G where there are exactly k non-optimal components. For any state $x \in \Omega$, define $H(x) = \mathbf{E}[H_x(Q_0)]$, i.e., the expected hitting time of an optimal state from x when running WalkSAT. Define $f_k = \min_{x \in Q_k} H(x)$; in particular, $f_0 = 0$, and f_1 corresponds to some state that differs from an optimal by only one bit. Define $g_k = f_{k+1} - f_k$. For any $x, y \in \Omega$, let $\Pr(x \to y)$ be the transition probability of WalkSAT, i.e., the probability that next state will be y given current state x. Note that $\Pr(x \to y) > 0$ only if $y \in N(x)$, where $N(x)$ is the set of states that differ from x by at most one bit. For any $A \subseteq \Omega$, define $\Pr(x \to A) = \sum_{y \in A} \Pr(x \to y)$.

For any $x \in Q_k$, we have

$$\begin{aligned} H(x) &= 1 + \sum_{y \in \Omega} \Pr(x \to y) H(y) \\ &= 1 + \sum_{t \in \{-1,0,1\}} \sum_{y \in Q_{k+t}} \Pr(x \to y) H(y) \\ &\geqslant 1 + \sum_{t \in \{-1,0,1\}} \sum_{y \in Q_{k+t}} \Pr(x \to y) f_{k+t}. \end{aligned}$$

Define

$$P^x_+ = \Pr(x \to Q_{k+1}), \quad P^x_- = \Pr(x \to Q_{k-1}),$$

then $\Pr(x \to Q_k) = 1 - P^x_+ - P^x_-$, and

$$H(x) \geqslant 1 + f_k(1 - P^x_+ - P^x_-) + f_{k-1}P^x_- + f_{k+1}P^x_+.$$

Since this inequality holds for any $x \in Q_k$, we can fix it to be some $x^* \in Q_k$ s.t. $H(x^*) = f_k$. Then $g_{k-1}P^{x^*}_- \geqslant 1 + g_k P^{x^*}_+$, which implies $g_{k-1} \geqslant g_k P^{x^*}_+ / P^{x^*}_-$.

Now without loss of generality assume that in x^*, $G_1, \ldots, G_k$ are non-optimal while $G_{k+1}, \ldots, G_N$ are optimal. Let x^*_i be the projection of x^* on G_i. Then since

$$P^{x^*}_- = \frac{\sum_1^k \nu_i(x^*_i)\alpha_i(x^*_i)}{\sum_1^N \nu_i(x^*_i)}, \quad P^{x^*}_+ = \frac{\sum_{k+1}^N \nu_j(x^*_j)\beta_j(x^*_j)}{\sum_1^N \nu_i(x^*_i)},$$

we have

$$g_{k-1} \geqslant g_k \frac{\sum_{k+1}^N \nu_j(x^*_j)\beta_j(x^*_j)}{\sum_1^k \nu_i(x^*_i)\alpha_i(x^*_i)} \geqslant g_k \frac{r(N-k)}{k},$$

where the second inequality follows from the definition of r.

For all $k \leqslant rN/(r+2)$, we have $g_{k-1} \geqslant 2g_k$. Since $g_k \geqslant 1$ for any k, $f_1 = g_0 \geqslant 2^{rN/(r+2)}$. That is, not aware of components, WalkSAT would take an exponential number of steps in expectation to correct the last bit to reach an optimum. □

According to this theorem, the gap on Example 5.1 is at least $2^{N/3}$; in fact, a more detailed analysis reveals that the gap is at least $\binom{N-1}{\frac{N}{2}} \approx \Theta(2^N/\sqrt{N})$. Figure A.1 shows the experiment results of running ALCHEMY, TUFFY, and TUFFY-p (i.e., TUFFY without partitioning) on Example 5.1 with 1000 components. Note that the analysis of Theorem 5.2 actually applies to not only WalkSAT, but stochastic local search in general. Since stochastic local search algorithms are used in many statistical models, we believe

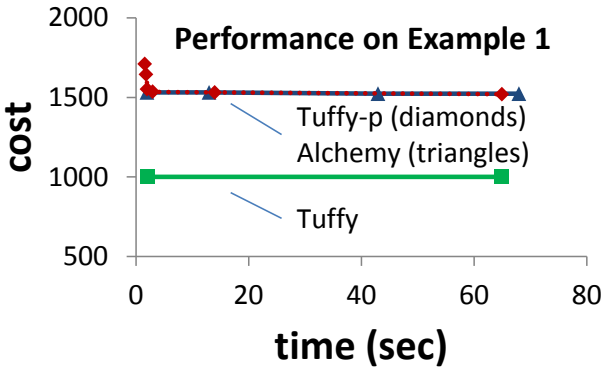


Figure A.1: *Effect of partitioning on Example 5.1*

that our observation here and corresponding techniques have much wider implications than MLN inference.

A.1.6 Hardness of MRF Partitioning

A bisection of a graph $G = (V, E)$ with an even number of vertices is a pair of disjoint subsets $V_1, V_2 \subset V$ of equal size. The cost of a bisection is the number of edges adjacent to both V_1 and V_2. The problem of *Minimum Graph Bisection* (MGB) is to find a bisection with minimum cost. This problem admits no PTAS [60]. The hardness of MGB directly implies the hardness of partitioning MRFs. As such, one may wonder if it still holds w.r.t. the domain size for a *given* MLN program (hence of size $O(1)$). The following theorem shows that the answer is yes.

Theorem A.2. *MGB can be reduced to the problem of finding a minimum bisection of the MRF generated an MLN of size* $O(1)$.

Proof. Consider the MLN that contains a single formula of the following form:

$$p(x), r(x, y) \rightarrow p(y),$$

where p is query and r is evidence. For any graph $G = (V, E)$, we can set the domain of the predicates to be V, and let $r = E$. The MRF generated by the above MLN is identical to G. □

A.1.7 MRF Partitioning Algorithm

We provide a very simple MRF partitioning algorithm (Algorithm 12) that is inspired by Kruskal's minimum spanning tree algorithm. It agglomeratively merges atoms into partitions with one scan of the clauses sorted in the (descending) absolute values of weights. The hope is to avoid cutting high-weighted clauses, thereby (heuristically) minimizing weighted cut size.

To explain the partitioning procedure, we provide the following definitions. Each clause c in the MRF $G = (V, E)$ is *assigned* to an atom in c. A partition of the MRF is a subgraph $G_i = (V_i, E_i)$ defined by a subset of atoms $V_i \subseteq V$; E_i is the set of clauses assigned to some atom in V_i. The size of G_i as referred to by Algorithm 12 can be any monotone function in G_i; in practice, it is defined to be the total number of literals and atoms in G_i. Note that when the parameter β is set to $+\infty$, the output is the connected components of G.

Algorithm 12 A Simple MRF Partitioning Algorithm

Input: an MRF $G = (V, E)$ with clause weights $w : E \mapsto \mathbb{R}$
Input: partition size bound β
Output: a partitioning of V s.t. the size of each partition is no larger than β

1: Initialize hypergraph $H = (V, F)$ with $F = \emptyset$
2: **for all** $e \in E$ in $|w|$-descending order **do**
3: $\quad F \leftarrow F \cup e$ if afterwards no component in H is larger than β
4: **return** the collection of per-component atom sets in H

Our implementation of Algorithm 12 only uses RAM to maintain a union-find structure of the nodes, and performs all other operations in the

RDBMS. For example, we use SQL queries to "assign" clauses to atoms and to compute the partition of clauses from a partition of atoms.

A.1.8 Tradeoff of MRF Partitioning

Clearly, partitioning might be detrimental to search speed if the cut size is large. Furthermore, given multiple partitioning options, how do we decide which one is better? As a baseline, we provide the following formula to (roughly) estimate the benefit (if positive) or detriment (if negative) of a partitioning:

$$W = 2^{\frac{N}{3}} - T\frac{|\#cut\_clauses|}{|E|},$$

where N is the estimated number of components with positive lowest cost, T is the total number of WalkSAT steps in one round of Gauss-Seidel, and $|E|$ is the total number of clauses. The first term roughly captures the speed-up as a result of Theorem 5.2, and the second term roughly captures the slow-down caused by cut clauses.

Empirically however, we find this formula to be rather conservative compared to experimental results that generally favor much more aggressive partitioning.

A.2 Additional Experiments

A.2.1 Alternative Search Algorithms

As shown in Section 5.3.3, RDBMS-based implementation of WalkSAT is several orders of magnitude slower than the in-memory counter part. This gap is consistent with the I/O performance of disk vs. main memory. One might imagine some clever caching schemes for WalkSAT, but even assuming that a flip incurs only one random I/O operation (which is usually on the order of 10 ms), the flipping rate of RDBMS-based search is

	LP	IE	RC	ER
Full optimizer	6	13	40	106
Fixed join order	7	13	43	111
Fixed join algorithm	112	306	>36,000	>16,000

Table A.1: *Grounding time in seconds*

	IE	RC
Tuffy-batch	448	133
Tuffy	117	77
Tuffy+parallelism	28	42

Table A.2: *Comparison of execution time in seconds*

still no more than 100 flips/sec. Thus, it is highly unlikely that disk-based search implementations could catch up to their in-memory counterpart.

A.2.2 Lesion Study of Tuffy Grounding

To understand which part of the RDBMS contributes the most to Tuffy's fast grounding speed, we conduct a lesion study by comparing the grounding time in three settings: 1) **full optimizer**, where the RDBMS is free to optimize SQL queries in all ways; 2) **fixed join order**, where we force the RDBMS to use the same join order as Alchemy does; 3) **fixed join algorithm**, where we force the RDBMS to use nested loop join only. The results are shown in Table A.1. Clearly, being able to use various join algorithms is the key to Tuffy's fast grounding speed.

A.2.3 Data Loading and Parallelism

To validate the importance of batch data loading and parallelism (Section 5.2.3), we run three versions of Tuffy on the IE and RC datasets: 1) **Tuffy**, which has batch loading but no parallelism; 2) **Tuffy-batch**,

which loads components one by one and does not use parallelism; and 3) **Tuffy+parallelism**, which has both batch loading and parallelism. We use the same WalkSAT parameters on each component (up to 10^6 flips per component) and run all three settings on the same machine with an 8-core Xeon CPU. Table A.2 shows the end-to-end running time of each setting.

Clearly, loading the components one by one incurs significant I/O cost on both datasets. The grounding + partitioning time of IE and RC are 11 seconds and 35 seconds, respectively. Hence, Tuffy+parallelism achieved roughly 6-time speed up on both datasets.

A.3 Additional Features in Tuffy

In addition to the syntax and semantics described in Section 2.2.3, from our experience of building KBC applications with Markov logic, we have found two language extensions to be instrumental to Tuffy (and Elementary): *scoping rules* and *parameterized weights*.

Scoping Rules By default, the arguments in an MLN predicate are considered to be independent to each other; i.e., for a predicate of arity a, there are $|D|^a$ possible ground tuples where D is the domain. Thus, given a query predicate `AdjacentMentions`(mention, mention), a typical MLN system – e.g., Alchemy[1] or Tuffy [87] – would enumerate and perform inference on #mentions^2 ground tuples instead of only the #mention-1 truly adjacent mention pairs. To address this issue, Elementary allows a developer to explicitly scope a query predicate with rules of the form

$$\texttt{AdjacentMentions}(m1, m2) :=$$
$$\texttt{Mentions}(m1, p1) \wedge \texttt{Mentions}(m2, p2) \wedge p1 = p2 - 1$$

[1] `http://alchemy.cs.washington.edu/`

where the evidence relation `Mentions` lists all mentions and their positional indexes (consecutive integers). ELEMENTARY implements scoping rules by translating them into SQL statements that are used to limit the content of the in-database representation of MLN predicates.

Parameterized Weights As we will see, an MLN can express classical statistical models such as logistic regression and conditional random fields [70]. Such classical statistical models usually have simple structure but many parameters. To compactly represent these models in MLN, we introduce *parameterized weights*, i.e., weights that are functions of one or more variables in an MLN rule. For example, a logistic regression model on the `TokenLabel` relation may have a parameter for each word-label pair; we can represent such a model using the following MLN rule

$$
\begin{aligned}
&wgt(w, l) : \\
&\texttt{Tokens}(\texttt{tok}) \wedge \texttt{HasWord}(\texttt{tok}, w) => \texttt{TokenLabel}(\texttt{tok}, l)
\end{aligned}
$$

where $wgt(w, l)$ is a parameterized weight that depends on the word variable w and the label variable l. Besides compactness, a key advantage of parameterized weights is that one can write a program that depends on the input data without actually providing the data. Another benefit is that of efficiency: a rule with parameterized weights may correspond to hundreds of thousands of MLN rules; because there can be non-negligible overhead in processing an MLN rule (e.g., initializing in-memory or in-database data structures), without this compact representation, we have found that the overhead of processing an MLN program can be prohibitive. ELEMENTARY implements parameterized weights by storing functions like $wgt(w, l)$ as look-up tables in a database that can be joined with predicate tables.

B FELIX

In this section, we formally describe the dual decomposition framework used in FELIX to coordinate the tasks. We start by formalizing MLN inference as an optimization problem. Then we show how to apply dual decomposition on these optimization problems.

Notation Used Table B.1 defines some common notation that is used in the following sections.

Notation	Definition
$a, b, \ldots, \alpha, \beta, \ldots$	Singular (random) variables
$\mathbf{a}, \mathbf{b}, \ldots, \boldsymbol{\alpha}, \boldsymbol{\beta}, \ldots$	Vectorial (random) variables
$\boldsymbol{\mu}' \cdot \boldsymbol{\nu}$	Dot product between vectors
$\lvert\boldsymbol{\mu}\rvert$	Length of a vector or size of a set
μ_i	i^{th} element of a vector
$\hat{\alpha}, \hat{\boldsymbol{\alpha}}$	A value of a variable

Table B.1: *Notation used in this section*

B.0.1 Problem Formulation

Suppose an MLN program Γ consists of a set of ground MLN rules $\mathcal{R} = \{r_1, ..., r_m\}$ with weights $(w_1, ..., w_m)$. Let $X = \{x_1, ..., x_n\}$ be the set of Boolean random variables corresponding to the ground atoms occurring in Γ. Each MLN rule r_i introduces a function ϕ_i over the set of random variables $\pi_i \subseteq X$ mentioned in r_i: $\phi_i(\pi_i) = 1$ if r_i is violated and 0 otherwise. Let $\boldsymbol{w}$ be a vector of weights. Define vector $\boldsymbol{\phi}(X) = (\phi_1(\pi_1), ..., \phi_m(\pi_m))$. Given a possible world $\mathbf{x} \in 2^X$, the cost is

$$\text{cost}(\mathbf{x}) = \boldsymbol{w} \cdot \boldsymbol{\phi}(\mathbf{x})$$

Suppose FELIX decides to decompose Γ into t tasks $O_1, ..., O_t$. Each task O_i contains a set of rules $\mathcal{R}_i \subseteq \mathcal{R}$. The set $\{\mathcal{R}_i\}$ forms a partition of $\mathcal{R}$. Let the set of random variables for each task be $X_i = \cup_{r_j \in \mathcal{R}_i} \pi_j$. Let $n_i = |X_i|$. Thus, each task O_i essentially solves the MLN program defined by random variables X_i and rules $\mathcal{R}_i$. Given $\boldsymbol{w}$, define $\boldsymbol{w}^i$ to be the weight vector whose entries equal $\boldsymbol{w}$ if the corresponding rule appears in $\mathcal{R}_i$ and 0 otherwise. Because $\mathcal{R}_i$ forms a partition of $\mathcal{R}$, we know $\sum_i \boldsymbol{w}^i = \boldsymbol{w}$. For each task O_i, define an n-dim vector $\boldsymbol{\mu}_i(X)$, whose j^{th} entry equals x_j if $x_j \in X_i$ and 0 otherwise. Define n-dim vector $\boldsymbol{\mu}(X)$ whose j^{th} entry equals x_j. Similarly, let $\boldsymbol{\phi}(X_i)$ be the projection of $\boldsymbol{\phi}(X)$ onto the rules in task O_i.

B.0.2 MAP Inference

MAP inference in MLNs is to find an assignment $\boldsymbol{x}$ to X that minimizes the cost:

$$\min_{\boldsymbol{x} \in \{0,1\}^n} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}). \tag{B.1}$$

Each task O_i performs MAP inference on X_i:

$$\min_{\boldsymbol{x}_i \in \{0,1\}^{n_i}} \boldsymbol{w}^i \cdot \boldsymbol{\phi}(\boldsymbol{x}_i). \tag{B.2}$$

Our goal is to reduce the problem represented by Eqn. B.1 into sub-problems represented by Eqn. B.2. Eqn. B.1 can be rewritten as

$$\min_{\boldsymbol{x} \in \{0,1\}^n} \sum_{1 \leqslant i \leqslant t} \boldsymbol{w}^i \cdot \boldsymbol{\phi}(\boldsymbol{x}_i).$$

Clearly, the difficulty lies in that, for $i \neq j$, X_i and X_j may overlap. Therefore, we introduce a copy of variables for each O_i: X_i^C. Eqn. B.1 now becomes:

$$\begin{aligned} \min_{x_i^C \in \{0,1\}^{n_i}, \boldsymbol{x}} \quad & \sum_i \boldsymbol{w}^i \cdot \boldsymbol{\phi}(\boldsymbol{x}_i^C) \\ \text{s.t.} \quad & \forall i \quad \boldsymbol{x}_i^C = \boldsymbol{x}. \end{aligned} \tag{B.3}$$

The Lagrangian of this problem is:

$$\begin{aligned} & \mathcal{L}(\boldsymbol{x}, \boldsymbol{x}_1^C, ..., \boldsymbol{x}_t^C, \boldsymbol{\nu}_1, ..., .\boldsymbol{\nu}_t) \\ = & \sum_i \boldsymbol{w}^i \cdot \boldsymbol{\phi}(\boldsymbol{x}_i^C) + \boldsymbol{\nu}_i \cdot (\boldsymbol{\mu}_i(\boldsymbol{x}_i^C) - \boldsymbol{\mu}_i(\boldsymbol{x})) \end{aligned} \tag{B.4}$$

Thus, we can relax Eqn. B.1 into

$$\max_{\boldsymbol{\nu}} \left\{ \sum_i \left[\min_{x_i \in \{0,1\}^{n_i}} \boldsymbol{w}^i \cdot \boldsymbol{\phi}(\boldsymbol{x}_i^C) + \boldsymbol{\nu}_i \cdot \boldsymbol{\mu}_i(\boldsymbol{x}_i^C) \right] - \max_{\boldsymbol{x}} \sum_i \boldsymbol{\nu}_i \cdot \boldsymbol{\mu}_i(\boldsymbol{x}) \right\}$$

The term $\max_{\boldsymbol{x}} \sum_i \boldsymbol{\nu}_i \cdot \boldsymbol{\mu}_i(\boldsymbol{x}) = \infty$ unless for each variable x_j,

$$\sum_{O_i : x_j \in X_i} \boldsymbol{\nu}_{i,j} = 0.$$

Converting this into constraints, we get

$$\begin{aligned} \max_{\boldsymbol{\nu}} \quad & \left\{ \sum_i \min_{x_i \in \{0,1\}^{n_i}} \boldsymbol{w}^i \cdot \boldsymbol{\phi}(\boldsymbol{x}_i^C) + \boldsymbol{\nu}_i \cdot \boldsymbol{\mu}_i(\boldsymbol{x}_i^C) \right\} \\ \text{s.t.} \quad & \forall x_j \quad \sum_{O_i : x_j \in X_i} \boldsymbol{\nu}_{i,j} = 0 \end{aligned}$$

We can apply sub-gradient methods on $\boldsymbol{\nu}$. The dual decomposition procedure in FELIX works as follows:

1. Initialize $\boldsymbol{\nu}_1^{(0)}, ..., \boldsymbol{\nu}_t^{(0)}$.

2. At step k (starting from 0):

a) For each task O_i, solve the MLN program consisting of: 1) original rules in this task, which are characterized by $\boldsymbol{w}^i$; 2) additional priors on each variables in X_i, which are characterized by $\boldsymbol{\nu}_i^{(k)}$.

b) Get the MAP inference results $\hat{\boldsymbol{x}}_i^C$.

3. Update $\boldsymbol{\nu}_i$:
$\boldsymbol{\nu}_{i,j}^{(k+1)} = \boldsymbol{\nu}_{i,j}^{(k)} - \lambda \left(\hat{\boldsymbol{x}}_{i,j}^C - \frac{\sum_{l:x_j \in X_l} \hat{\boldsymbol{x}}_{l,j}^C}{|\{l:x_j \in X_l\}|} \right)$

B.0.3 Marginal Inference

Marginal inference of MLNs aims at computing the marginal distribution (i.e., the expectation since we are dealing with boolean random variables):

$$\hat{\boldsymbol{\mu}} = \mathbb{E}_{\boldsymbol{w}}[\boldsymbol{\mu}(X)]. \tag{B.5}$$

The sub-problem of each task is of the form:

$$\hat{\boldsymbol{\mu}}_O = \mathbb{E}_{\boldsymbol{w}_O}[\boldsymbol{\mu}_O(X_O)]. \tag{B.6}$$

Again, the goal is to use solutions for Eqn. B.6 to solve Eqn. B.5.

We first introduce some auxiliary variables. Recall that $\boldsymbol{\mu}(X)$ corresponds to the set of random variables, and $\boldsymbol{\phi}(X)$ corresponds to all functions represented by the rules. We create a new vector $\boldsymbol{\xi}$ by concatenating $\boldsymbol{\mu}$ and $\boldsymbol{\phi}$: $\boldsymbol{\xi}(X) = (\boldsymbol{\mu}^T(X), \boldsymbol{\phi}^T(X))$. We create a new weight vector $\boldsymbol{\theta} = (0, ..., 0, \boldsymbol{w}^T)$ which is of the same length as $\boldsymbol{\xi}$. It is not difficult to see that the marginal inference problem equivalently becomes:

$$\hat{\boldsymbol{\xi}} = \mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\xi}(X)]. \tag{B.7}$$

Similarly, we define $\boldsymbol{\theta}_O$ for task O as $\boldsymbol{\theta}_O = (0, ..., 0, \boldsymbol{w}_O^T)$. We also define a set of $\boldsymbol{\theta}$: Θ_O, which contains all vectors with entries corresponding to

random variables or cliques not appear in task O as zero. The partition function $A(\theta)$ is:

$$A(\theta) = \sum_{\mathcal{X}} \exp\{-\theta \cdot \xi(\mathcal{X})\}$$

The conjugate dual to A is:

$$A^*(\xi) = \sup_{\theta}\{\theta \cdot \xi - A(\theta)\}$$

A classic result of variational inference [137] shows that

$$\hat{\xi} = \arg\sup_{\xi \in \mathcal{M}}\{\theta \cdot \xi - A^*(\xi)\}, \tag{B.8}$$

where $\mathcal{M}$ is the marginal polytope. Recall that $\hat{\xi}$ is our goal (see Eqn. B.7). Similar to MAP inference, we want to decompose Eqn. B.8 into different tasks by introducing copies of shared variables. We first try to decompose $A^*(\xi)$. In $A^*(\xi)$, we search θ on all possible values for θ. If we only search on a subset of θ, we can get a lower bound:

$$A^{*O}(\xi) = \sup_{\theta \in \Theta_O}\{\theta \cdot \xi - A^*(\xi)\} \leqslant A^*(\xi).$$

Therefore,

$$-A^*(\xi) \leqslant \frac{1}{m}\sum_{O} -A^{*O}(\xi),$$

where m is the number of tasks. We approximate $\hat{\xi}$ using this bound:

$$\hat{\xi} = \arg\sup_{\xi \in \mathcal{M}}\{\theta \cdot \xi - \frac{1}{m}\sum_{O} A^{*O}(\xi)\},$$

which is an upper bound of the original goal. We introduce copies of ξ:

$$\hat{\xi} = \arg \sup_{\xi^{O_i} \in \mathcal{M}, \xi} \{\sum_{O} \theta_O \cdot \xi^O - \frac{1}{m} \sum_{O} A^{*O}(\xi^O)\}$$
$$\text{s.t. } \xi^O_e = \xi_e, \forall e \in \mathcal{X}_O \cup \mathcal{R}_O, \forall O$$

The Lagrangian of this problem is:

$$\mathcal{L}(\xi, \xi^{O_1}, ..., \xi^{O_t}, \nu_1, ..., \nu_t) = \sum_{O} \left\{ \theta_O \cdot \xi^O - \frac{1}{m} A^{*O}(\xi^O) \right\} + \sum_{i} \nu_i \cdot (\xi^{O_i} - \xi),$$

where $\nu_i \in \Theta_i$, which means only the entries corresponding to random variables or cliques that appear in task O_i are allowed to have non-zero values. We get the relaxation:

$$\min_{\nu_i \in \Theta_i} \sum_{i} \sup_{\xi^{O_i} \in \mathcal{M}} \left\{ \theta_i \cdot \xi^{O_i} - \frac{1}{m} A^{*O_i}(\xi^{O_i}) + \nu_i \cdot \xi^{O_i} \right\} - \min_{\xi} \sum_{i} \nu_i \cdot \xi$$

Considering the $\min_\xi \sum_i \nu_i \cdot \xi$ part. This part is equivalent to a set of constraints:

$$\sum_{O_i : x \in X_i} \nu_{i,x} = 0, \forall x \in X$$
$$\nu_{i,x} = 0, \forall x \notin X$$

Therefore, we are solving:

$$\min_{\nu_i \in \Theta_i} \sum_{i} \sup_{\xi^{O_i} \in \mathcal{M}} \left\{ m\theta_i \cdot \xi^{O_i} - A^{*O_i}(\xi^{O_i}) + \nu_i \cdot \xi^{O_i} \right\}$$
$$\text{s.t., } \sum_{O_i : x \in X_i} \nu_{i,x} = 0, \forall x \in X$$
$$\nu_{i,x} = 0, \forall x \notin X$$

Note the factor m in front of θ_i; it implies that we multiply the weights in each subprogram by m as well. Then we can apply sub-gradient method on $\boldsymbol{\nu}_i$:

1. Initialize $\boldsymbol{\nu}_1^{(0)}, ..., \boldsymbol{\nu}_t^{(0)}$.
2. At step k (start from 0):
   a) For each task O_i, solve the MLN program consists of: 1) original rules in this task, which is characterized by $m\theta_i$; 2) additional priors on each variables in $\mathcal{X}_i$, which is characterized by $\boldsymbol{\nu}_i^{(k)}$.
   b) Get the marginal inference results $\hat{\xi}_i^{C}$.
3. Update $\boldsymbol{\nu}_i^{(k+1)}$:
   $\boldsymbol{\nu}_{i,j}^{(k+1)} = \boldsymbol{\nu}_{i,j}^{(k)} - \lambda \left(\hat{\xi}_{i,j}^{C} - \frac{\sum_{l:x_j \in X_l} \hat{\xi}_{l,j}^{C}}{|\{l:x_j \in X_l\}|} \right)$