ABSTRACT

Title of dissertation: PROBABILISTIC MODELS FOR SCALABLE KNOWLEDGE GRAPH CONSTRUCTION

Jay Pujara,
Doctor of Philosophy, 2016

Dissertation directed by: Professor Lise Getoor
Department of Computer Science

In the past decade, systems that extract information from millions of Internet documents have become commonplace. Knowledge graphs – structured knowledge bases that describe entities, their attributes and the relationships between them – are a powerful tool for understanding and organizing this vast amount of information. However, a significant obstacle to knowledge graph construction is the unreliability of the extracted information, due to noise and ambiguity in the underlying data or errors made by the extraction system and the complexity of reasoning about the dependencies between these noisy extractions. My dissertation addresses these challenges by exploiting the interdependencies between facts to improve the quality of the knowledge graph in a scalable framework. I introduce a new approach called *knowledge graph identification* (KGI), which resolves the entities, attributes and relationships in the knowledge graph by incorporating uncertain extractions from multiple sources, entity co-references, and ontological constraints. I define a probability distribution over possible knowledge graphs and infer the most probable knowledge graph using a combination of probabilistic and logical reasoning. Such probabilistic mod-

els are frequently dismissed due to scalability concerns, but my implementation of KGI maintains tractable performance on large problems through the use of hinge-loss Markov random fields, which have a convex inference objective. This allows the inference of large knowledge graphs using 4M facts and 20M ground constraints in 2 hours. To further scale the solution, I develop a distributed approach to the KGI problem which runs in parallel across multiple machines, reducing inference time by 90%. Finally, I extend my model to the streaming setting, where a knowledge graph is continuously updated by incorporating newly extracted facts. I devise a general approach for approximately updating inference in convex probabilistic models, and quantify the approximation error by defining and bounding inference regret for online models. Together, my work retains the attractive features of probabilistic models while providing the scalability necessary for large-scale knowledge graph construction. These models have been applied on a number of real-world knowledge graph projects, including the NELL project at Carnegie Mellon and the Google Knowledge Graph.

PROBABILISTIC MODELS FOR SCALABLE KNOWLEDGE GRAPH CONSTRUCTION

by

Jay Pujara

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2016

Advisory Committee:

Professor Lise Getoor, University of Maryland, Chair
Professor William W. Cohen, Carnegie Mellon University
Professor Hector Corrada-Bravo, University of Maryland
Professor Hal Daumé III, University of Maryland
Prof. Philip Resnik, University of Maryland

Dedication

Thats it. The lover writes, the believer hears,
The poet mumbles and the painter sees,
Each one, his fated eccentricity,
As a part, but part, but tenacious particle,
Of the skeleton of the ether, the total
Of letters, prophecies, perceptions, clods
Of color, the giant of nothingness, each one
And the giant ever changing, living in change.
–Wallace Stevens, from *A Primitive Like an Orb*

Acknowledgments

As a lover of wordplay, I would sometimes imagine that scholarship was, instead of a task undertaken in the windowless offices of the AV Williams building, an actual ship – a tall schooner, plowing through the dark, mysterious seas of Research with its crisp, white sails billowing in the wind. However, if the metaphor has any truth to it, this ship would have one (or possibly many) leaks of unknown origin, the lines would be tangled in an irredeemable mess, and it would be well off course and weeks behind schedule. Amidst those sinking realizations, I would often temper my despair by reflecting how thankful I was that I was not on this ship alone.

My advisor Lise Getoor deserves most of the credit for keeping me from running aground. I can't say why she thought I was a good bet when I arrived at her office in March 2010 after crossing a continent and navigating the Metrobus system: sweaty, unkempt, carrying a 90 liter hiking pack and babbling disconnectedly about half-baked research ideas. Whatever her reasons, I'm extremely thankful to have the opportunity to work with her. Lise has been patient, allowing me the time to explore many avenues that turned out to be dead ends and forgiving many late weekly reports; encouraging, battling my skepticism and pessimism when confronted with frustrating datasets or bleak experimental results; and dedicated, investing her nights and weekends to sending meticulously (and often distressingly) annotated paper drafts in hopes my writing would improve. Above all, I have been astounded by how much Lise cares about her students and how hard she works to help them achieve their goals, whatever they may be. Lise has also helped me forge connections with many of the mentors who have helped me through the course of my studies.

One of these mentors is William Cohen, who was critical in shaping much of my work on knowledge graphs. I benefited from William's deep experience in NLP and machine learning, his pragmatic approach to slashing through the most troubling Gordian knots, and his ability to connect me with just the right person to solve my data woes. As a rule, no matter what problem I brought into a meeting with William, by the end he would have pointed me in the right direction to make progress. I'm grateful to have had the opportunity of this collaboration, as well as the opportunity to work with William briefly at Carnegie Mellon.

I also owe a debt of gratitude to Hal Daumé III. Hal joined the University of Maryland at the same time as me (albeit as a professor), and his office was directly across from the LINQS lab so I've been bugging him throughout my entire graduate career. I'm not sure how to describe Hal's magic, but every time I walked out of his office I would feel better, whether it was because I finally understood the optimization problem I'd been working on for a week, or had a new perspective on the related work for an idea I'd encountered, or simply because he helped defuse my anxiety and gave me fresh hope.

Many of my most formative experiences as a researcher have been due to all I have learned as a member of the LINQS lab. I can't enumerate all of the ways being a member

of LINQS has contributed to my maturation as a researcher, but it spans everything from the weekly reading group to nightly Skeeball sessions in South Lake Tahoe during the NIPS conferences. While what I've learned is hard to list, the wonderful people I've had the opportunity to learn from comes readily to mind. I'd like to thank the various brilliant postdocs who have been part of the group and whose particular expertise I've benefited from: Jimmy Foulds, Bert Huang, Angelika Kimmig, Stanley Kok, and Lily Mihalkova. When I started in LINQS, I was extremely appreciative of the sage advice and guidance of the senior students: Mustafa Bilgic, Matthias Broecheler, Walaa Eldin Moustafa, Galileo Namata, Hossam Sharara, and Elena Zheleva. Three other LINQS members joined the lab the same time I did: Steve Bach, Ben London, and Theo Rekatsinas. I cannot express how much their friendship, guidance, and help has meant to me. LINQS has also had a number of visiting students in various forms, with whom I've had fruitful collaborations: Golnoosh Farnadi, Adam Grycner, Eric Norris, and Natalia Diaz Rodriguez. Finally, I want to thank the next generation of LINQS, who have taught me so much and been incredibly kind and welcoming: Shobeir Fakhrei, Matthew Howard, Pigi Kouki, Alex Memory, Hui Miao, Arti Ramesh, Dhanya Sridhar, and Sabina Tomkins – I'm glad to see them continuing the vibrancy of the LINQS community.

The work in this dissertation would have been impossible without help from a number of people whose help at critical points greatly influenced the direction of my research. I'd like to thank the NELL group at Carnegie Mellon, particularly Bryan Kisiel, Jayant Krishnamurthy, Bhavana Dalvi Mishra, and Anthony Platanios. Serendipitously, working on NELL allowed me to reconnect with Tom Mitchell, who was my advisor as an undergraduate and Masters student at Carnegie Mellon, and gave me the taste for research which brought me back to my PhD. I'm grateful to Hector Corrada Bravo and Philip Resnik for their diligent service on my committee. I'd also like to give special thanks to Hui Miao, who provided crucial help during my initial explorations of KGI, and Ben London, who helped me distill my amorphous ideas of how streaming inference should work into workable theory.

I've have had some wonderful mentors at two internships during my graduate studies. While at Google, I had the great fortune to work with Luna Dong, Evgeniy Gabrilovich, Curtis Janssen, Kevin Murphy, Wei Zhang, and other members of the Knowledge Vault team. Their support helped me work with the vast real data available at Google and mater the necessary tools. Similarly, I had an excellent time working at LinkedIn, enabled with the support of Mathieu Bastian, Christopher Lloyd, Pete Skomoroch, Sal Uryasev, William Vaughan and the many other generous scientists in Decision Sciences. I would also like to thank my fellow interns who helped me along the way, in particular: Tim Althoff, Praveen Bommannavar, Arun Chaganty, Ari Kobren, Matthieu Monsch, and Karthik Raman.

The ties between the many people I've thanked above and my research is fairly direct; I can recall anecdotes about how their help has advanced my ability as a researcher. However, the key ingredient that allowed me to complete a PhD is the support of my friends and family. I can't do justice to all of the people who have been there for me over the last six years, but I would like to acknowledge at least some of these special people. First, I want to thank Patrick Barry, Matt Brown, Anna Geisler, Brian Goodman, Neil Halelamien, Jonathan Hsu, Emily Huang, Karen Knee, Derek Leung, Jennifer Lin,

Danielle Little, Steve and Jane Onorato, Devesh and Masumi Parekh, Kris Popendorf, Deanna Rubin, Mark Schmit, Vaughn Tan, and Philip Yam – I've known most of this crowd over a decade and despite the separation of time and distance, I'm grateful we've remained connected. One of the intimidating aspects of starting a PhD is often moving somewhere new and establishing a new social circle. I've been lucky enough to find many great friends over my PhD, but the core circle of friends most responsible for my sanity by way of spontaneous parties, board games, and trivia night deserves special mention: Stephen Bach, Cody Buntain, Leigh Cook, Philip and Robin Dasler, Alexis Evangelos, Julian Grizzard, Renee LaGue, Steven Lee, Piotr Madrizel, Sana Malik, Matthew Mauriello, Alex Malozemoff, Brenna McNally, Kris Micinski, Issac Roberts, Karla Saur, and Amanda Strickler – I couldn't have done it without you, especially Renee. I'd also like to thank Brad Dettmer, Nicole Mendoza, Subhodeep Mishra, Soja-Marie Morgens, Dhanya Sridhar, and Rob Sumner who introduced me to many wonderful people during my travels to Santa Cruz and Pittsburgh.

Finally, the deepest support I've received comes from my family. My siblings, Shreya and Hirsh, have been incomparable companions, whether tempting the jaws of a petrified alligator on a hike or playing board games in front of a roaring fire or attempting a baking adventure. I can't imagine where I would be without the love and understanding of my parents Kirit and Smita; they've always been there for me, supported me in whatever I've chosen to undertake, and while they were frequently mystified by what I was doing and why it was taking so long, they never doubted I would succeed (although often doubted that I was eating a healthy dinner).

Portions of this work were supported by the National Science Foundation (NSF), under NSF CAREER grant 0746930 and NSF grant numbers CCF0937094 and IIS1218488; by the Intelligence Advanced Research Projects Activity (IARPA), via Department of Interior National Business Center (DoI/NBC) contract number D12PC00337; by the Air Force Research Lab (AFRL) contract #FA8750-10-C-0191; and by research grants from Google and Yahoo! The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon.

Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, IARPA, DoI/NBC, AFRL, Google, Yahoo! or the U.S. Government.

Table of Contents

1 Introduction 1
- 1.1 Opportunities . . . 2
- 1.2 Challenges . . . 4
- 1.3 Approach and Contributions . . . 5

2 Related Work 8
- 2.1 The Quest for Knowledge in AI . . . 9
- 2.2 Knowledge Representation and Reasoning . . . 10
- 2.3 Semantic Web: Ontologies and Tools . . . 12
- 2.4 Information and Knowledge Extraction . . . 13
- 2.5 Probabilistic Graphical Models and Structured Prediction . . . 16
- 2.6 Contemporary Approaches to Knowledge Base Construction . . . 17
- 2.7 Streaming and Online Inference . . . 19

3 Problem Formulation for Knowledge Graph Identification 22
- 3.1 Knowledge Graphs . . . 22
- 3.2 Common Errors in Information Extraction . . . 25
  - 3.2.1 Entity Ambiguity . . . 25
  - 3.2.2 Attribute Errors . . . 26
  - 3.2.3 Relation Extraction Errors . . . 28
- 3.3 Graph Identification . . . 28
- 3.4 Adapting Graph Identification to Knowledge Graphs . . . 30

4 Modeling Knowledge Graph Identification 32
- 4.1 Background: PSL for Knowledge Graphs . . . 32
- 4.2 Model for Knowledge Graph Identification . . . 35
  - 4.2.1 Representing Uncertain Extractions . . . 36
  - 4.2.2 Entity Resolution . . . 37
  - 4.2.3 Enforcing Ontological Constraints . . . 38
- 4.3 Implementing Knowledge Graph Identification with PSL . . . 39
- 4.4 Experimental Evaluation . . . 40
  - 4.4.1 Datasets and Experimental Setup . . . 40

4.4.2 Learning Model Weights from Training Data . . . 45
4.4.3 Open-World vs Closed-World Evaluation Setting . . . 47
4.4.4 Results for Closed-World Settings . . . 48
4.4.5 Results for Model Ablation Study . . . 52
4.4.6 Results for Open-World Settings . . . 56
4.5 Discussion . . . 58

5 Entity Resolution for Knowledge Graphs 60
5.1 Problem Definition . . . 60
5.1.1 Ambiguity In Candidate Extractions . . . 62
5.1.2 Incorporating New Extractions Into a Knowledge Graph . . . 63
5.1.3 Combining Information From Multiple Knowledge Graphs . . . 64
5.2 Approach . . . 65
5.2.1 Local and Collective Knowledge Graph Features . . . 65
5.2.2 Knowledge Graph Models at Different Granularity . . . 67
5.3 Modeling Knowledge Graph Entity Resolution . . . 68
5.3.1 Basic Features . . . 68
5.3.2 New Entity Features . . . 70
5.3.3 Abstract Knowledge Graph Features . . . 71
5.3.4 Domain-Specific Knowledge Graph Features . . . 73
5.3.5 Synthesis . . . 77
5.4 Evaluation . . . 78
5.5 Discussion . . . 82

6 Scaling Knowledge Graph Identification 84
6.1 Scalability Analysis of Knowledge Graph Identification . . . 84
6.2 Scaling Knowledge Graph Identification with HL-MRFs . . . 86
6.3 Scalability Challenges for Knowledge Graph Identification . . . 87
6.3.1 Partitioning Knowledge Graphs for Distributed Processing . . . 88
6.3.2 Scalability via Ontological Partitioning . . . 89
6.3.2.1 Handling Unevenly Distributed Extractions . . . 91
6.3.2.2 Dealing with Unbalanced Ontologies . . . 93
6.4 Evaluation . . . 94
6.4.1 Comparison of Partitioning Techniques . . . 95
6.4.2 Assessing the Impact of Partition Size . . . 97
6.5 Discussion . . . 99

7 Online Collective Inference 101
7.1 Preliminaries . . . 103
7.2 Inference Regret . . . 106
7.2.1 Regret Bounds for Strongly Convex Inference . . . 108
7.2.2 The Lipschitz Constant of the Features . . . 114
7.3 Algorithms for Online Inference Activation . . . 115
7.3.1 Background: ADMM Optimization . . . 115
7.3.2 ADMM Features . . . 117

7.3.3 Activation Algorithms . . . 118
7.4 Evaluation . . . 121
7.4.1 Online Collective Classification . . . 122
7.4.2 Collaborative Filtering . . . 125
7.5 Discussion . . . 127

8 Conclusion and Future Work 129
8.1 Future Work . . . 130

A Sample PSL Program for Knowledge Graph Identification 133

B Additional Results for Knowledge Graph Identification 139
B.1 Baseline Results . . . 139
B.2 Results Excluding Extractor Source Information . . . 142
B.3 Results Excluding Entity Resolution Information . . . 144
B.4 Results Excluding Ontological Information . . . 147
B.5 Results for the Knowledge Graph Identification Model . . . 150
B.6 Results for the Open-World Knowledge Graph Identification Model . . . 152

Bibliography 157

Chapter 1: Introduction

Knowledge has always been an essential ingredient in the quest to build intelligent agents and systems. Representing information about the world, reasoning about new or unobserved facts, and learning from the environment are key facets of intelligent behavior. As a result, the problems of representation, reasoning, and learning are among the defining challenges of artificial intelligence. Decades of research have lead to significant advances in each of these areas, and increasingly sophisticated approaches to collecting, organizing, and employing knowledge.

Concurrently, there has been an explosion of easily accessible machine-readable data and tools to process this data. For example, trillions of web pages and billions of videos are now available on the World Wide Web (WWW). Diverse, open-source toolkits are available to parse and understand text, perform voice recognition on audio, and recognize people and objects in video. The proliferation of publicly available information and powerful tools to extract this information have created myriad opportunities, particularly for creating systems that construct knowledge bases that span diverse domains.

However, the problem remains far from solved; many obstacles hinder the usefulness of knowledge base construction systems, and the resulting knowledge bases are often hampered by quality and coverage issues. In this dissertation, I explore the capabilities

and limitations of current knowledge base construction systems, enumerate some of the salient challenges that confront such systems, and develop models to address these challenges.

1.1 Opportunities

The past two decades have seen enormous changes in the landscape of how information is organized, accessed and processed. In contrast to twenty years ago, much of the information in the world is available in digital form. Content spanning newspapers, periodicals, books, scholarly writings and presentations, musical works, product and business information, instructional videos, television programs and movies are in digital, machine-readable formats. Moreover, many personal experiences are also recorded digitally, with photos, videos, and anecdotes shared using Internet technologies. The advent and popularity of the World Wide Web (WWW) and the ubiquity of personal computing devices has allowed billions of users access to this vast repository of publicly available information.

Simultaneously, academic research has made significant strides in two critical areas: extracting structured and semantically meaningful information from data and relating this information to rigorously-defined ontological formalisms. The former has been the work of the natural language and information extraction communities, and the latter due to the Semantic Web movement. Together, these advances provide the ingredients for practical knowledge base construction.

The first of these ingredients is a set of increasingly powerful tools to understand

texts and extract information using these features. The natural language processing community has produced better mechanisms for the key tasks in understanding text: parsing, part-of-speech tagging, named entity recognition, and semantic role labeling. These tasks yield the features that enable extracting entities and meaningful relationships between these entities. Information extraction research has devised a number of ways to use these features, along with modest amounts of training data, to extract a staggering number of entities and relationships from text.

Separately, the Semantic Web movement has worked to standardize knowledge representation, and provide tools to specify semantically meaningful and interoperable interpretations for information. A key effort of this community has been the definition of ontologies that serve as the schemas for knowledge. This technology allows relations and attributes in knowledge bases to be well-defined, and provides a set of constraints that ensure the coherency of the knowledge base. These tools have enabled data to be annotated with general and domain-specific ontological information.

The combination of data, algorithms, standards and systems together provide a tremendous opportunity for AI research to convert information into knowledge. Building on work on knowledge representation, the output of these systems holds the potential to overcome the knowledge acquisition bottleneck. By realizing the pursuit of knowledge, information retrieval systems will be able to go beyond simply indexing the documents on the WWW, and instead build web-scale systems capable of understanding the vast troves of information available on the Internet.

These developments also reflect the growing needs of the billions of people seeking knowledge. Currently every major search engine surfaces structured knowledge for

select queries, and every major mobile computing platform includes a digital assistant feature that attempts to interactively answer user queries with information from a knowledge base. Although these systems currently rely heavily on curated knowledge, the pace at which data is increasing makes the automatic construction of knowledge bases an inevitable necessity.

1.2 Challenges

Although automated knowledge base construction is a growing necessity, addressing this need poses many challenges. While there is an incredible amount of data available, the quality of this data varies widely. The vast majority of available data is not annotated with ontological information, and in some cases the annotations are erroneous. Data that lacks annotation may also contain errors: the information may be outdated, incorrect, or malicious. Even when reliable data is available, an information extraction system may make errors. These many sources of erroneous information can compromise a knowledge base.

Erroneous information extracted from data is a serious problem, but the preponderance of data also provides a source of robustness. One hope is that by using ontological knowledge and combining information from many sources, noise in the knowledge base can be resolved allowing us to produce a coherent knowledge base. However, this solution comes with its own challenge: scalability. Extraction systems can produce trillions of candidate facts, and ontological constraints and knowledge can introduce dependencies between any pair of facts. Resolving errors in a knowledge base requires considering the

myriad dependencies between facts.

Finally, even if a system is capable of processing the deluge of facts and dependencies between them, new information is constantly being extracted. A practical knowledge base system will have to grow and adapt as new information is introduced. However, if the knowledge base construction is prohibitively costly, incorporating new information may be impossible. Thus, a knowledge base construction requires a system that can efficiently update a knowledge base while still remaining robust to errors.

The challenge facing knowledge base construction efforts can be distilled into a simple requirement: a successful system must be capable of applying the depth and complexity of ontological knowledge to the wealth of data generated by modern information extraction approaches. Meeting this requirement requires walking a careful balance – the system must handle the statistical features from information extraction as well as the semantic constrains from ontologies while retaining scalable performance to deal with the large amount of data. This is the challenge I address in this dissertation.

1.3 Approach and Contributions

My work addresses the challenges confronting knowledge base construction. In this dissertation, I develop a system that is capable of surmounting noisy source information by incorporating dependencies between facts while remaining scalable, even in a streaming setting. The structure of this document mirrors the major contributions of my work, portions of which have been published elsewhere (Pujara et al., 2013a,b,c, 2014, 2015a,b).

In Chapter 3, I provide examples of errors found in knowledge bases. I identify the

common trends in these failures. I demonstrate that by representing the candidate facts of a knowledge base as a graph, correcting these errors correspond to the principal tasks in graph identification. I introduce a model, knowledge graph identification (KGI), that is capable of resolving errors in extracted knowledge bases. At the end of the chapter, I pose a hypothesis that KGI will improve the quality of extracted knowledge bases.

I test the hypothesis that knowledge graph identification improves knowledge base quality in Chapter 4. I first develop a model for knowledge graph identification that incorporates uncertain extractions and ontological knowledge. I implement this model as a probabilistic graphical model, and discuss the important design constraints for this model. I perform extensive experiments to assess the impact of my model design and validate the KGI hypothesis.

In Chapter 5, I extend knowledge graph identification to address one of the foremost challenges in knowledge graph construction: entity resolution. I perform an analysis of the entity resolution problem settings found in knowledge graphs. I identify and formalize the important features for entity resolution in knowledge graphs. Using these features, I develop a general, probabilistic approach to entity resolution that addresses the differing requirements of each problem setting. I implement my entity resolution model for two different problem settings to demonstrate its generality. In empirical evaluation, I show the power of my general entity resolution framework across problem settings.

A looming challenge in any discussion of knowledge base construction is scalability. In Chapter 6, I explicate the scalability challenges of knowledge graph identification by performing a complexity analysis for the core problems of KGI. I support the scalability of my implementation of KGI with theoretical analysis and empirical investigation

across different problem sizes. I propose a mechanism to improve scalability by distributing KGI across multiple machines. I implement a parallel KGI system, and demonstrate that this approach can yield ten-fold improvements in running time without significant loss of quality.

Another significant obstacle for practical knowledge base construction is the necessity of constantly updating the knowledge base as new information is extracted. In Chapter 7, I address the problem of growing and extending an existing knowledge graph in response to a stream of new extractions. I first formulate the general problem of collective online inference for probabilistic graphical models. To measure the performance of an online inference, I introduce a new error measure, inference regret. I develop a bound for inference regret in a regime where inference output is partially updated with new evidence. I invent a set of algorithms for updating inference that use features from the inference optimization. In empirical evaluation show that these algorithms reduce inference regret while maintaining model performance.

Chapter 2: Related Work

A number of research areas are related to the work I have undertaken. First, there is a significant body of work on knowledge representation and reasoning which has enabled powerful mechanisms for manipulating knowledge. The application of that work is most clearly seen in the work of the Semantic Web community, which has defined formalisms and built tools that have many practical applications in knowledge base construction. In parallel to work on knowledge representation is work on knowledge extraction – processing the raw data that can eventually be transformed into knowledge. A number of other knowledge base construction projects also build on knowledge extraction systems, although with very different approaches. One particular subproblem in knowledge base construction, entity resolution, is the subject of decades of research. Orthogonal to the many methods relevant to knowledge base construction is the question of scalability. A number of advances in scalable optimization and distributed systems have made large-scale knowledge graph construction feasible. Finally, one of the key properties of real-world knowledge graph settings is that the construction task is online and must be updated to incorporate a stream of new evidence. Several research projects have considered updating inference in probabilistic models, and there are a diverse set of projects that seek to model dynamic data that I cover briefly.

2.1 The Quest for Knowledge in AI

The goal of knowledge base construction has been a key aspect of AI research since its foundation. Among the earliest work in AI was the introduction of reasoning systems that used knowledge bases to prove or disprove query assertions (Russell and Norvig, 1995). One notable project in this pattern was the General Problem Solver (GPS) by Newell et al. (1959), which took as input an abstract set of knowledge that consisted of logical formulas and axioms, then used a search-based reasoning approach to decide whether a given logical statement was true or false. While the generality of GPS was among its attractive characteristics, this generality also constituted a significant weakness: search-based reasoning encountered the combinatorial explosion of exponentially many possible statements (Boden, 2008).

An ongoing theme of the subsequent work in knowledge base construction and reasoning has been addressing the problems of the combinatorial explosion while still retaining the power of knowledge-based systems. One development that garnered particular attention in the domain was the SHRDLU system (Winograd, 1972) which emphasized procedural knowledge over logical representation and restricted the domain to a *micro-world*, which offered a simplified environment with a limited set of objects, properties, and transformations. By enforcing these limitations, the SHRDLU system was able to complete a broad and versatile set of tasks in a block world environment while sidestepping the combinatorial explosion by restricting the size of the problem space.

SHRDLU and similar projects ushered in an era of knowledge-based approaches that took the form of *expert systems* (Buchanan and Shortliffe, 1984). Expert systems

focused on narrow domains and used knowledge bases and rules that were hand-crafted by experts (Russell and Norvig, 1995). Early successes such as DENDRAL (Buchanan and Sutherland, 1968; Feigenbaum et al., 1970; Buchanan and Feigenbaum, 1978), an expert system on for chemistry, and MYCIN (Shortliffe, 1974, 1976), an expert system in the medical domain, channeled the efforts of AI researchers towards expert systems for decades. While expert systems were often applied to limited domains, the approach also emboldened new work into commonsense reasoning and general-purpose knowledge bases, most vividly in the Cyc project.

The Cyc project (Lenat et al., 1985; Lenat and Guha, 1990; Lenat, 1995) took an expert-system approach by meticulously curating knowledge and crafting a deep and complex ontology for all knowledge. Operating over two decades, the project has amassed millions of assertions for hundreds of thousands of entities. One drawback and frequent criticism of this approach, and expert systems in general, is the *knowledge acquisition bottleneck*, describing the limitations of any computational approach that requires human intervention to operate (Lenat et al., 1985; Wagner, 2006).

2.2 Knowledge Representation and Reasoning

The diversity of approaches toward knowledge and reasoning in the AI community was accompanied by a diversity in the choices of knowledge representation (Barr and Davidson, 1981). The characteristics of these differing approaches to knowledge representation are shaped by a number of central choices. A principal question is whether knowledge should be represented explicitly, as semantically meaningful sentences, or implicitly, as

part of the reasoning procedure or predictive mechanism. A second concern is the scope of the representation: should the representation of knowledge be universal or framed by the context. A third essential question is how a representation can handle uncertainty in cases where background knowledge or predicted outcomes cannot be clearly judged to be true or false. A full exploration of these choices is beyond the scope of this chapter, but I present some of the major relevant arguments in this longstanding discussion.

A forceful set of arguments has supported the role of knowledge representations based on declarative logical forms. (Hayes, 1977; Nilsson, 1982) emphasizes the representational power of logic, argues that many other approaches (such as Semantic Networks and frame-based reasoning) are equivalent or inferior in their representation of knowledge, and dispels confusions about logic-based approaches. (Nilsson, 1991) also argues in support of logic-based artificial intelligence, with the proviso that intelligent machines will represent knowledge declaratively, using first-order predicate calculus. A key development which shaped logical approaches to defining and expanding knowledge was the introduction and refinement of description logics.

Description logics (Krötzsch et al., 2014; Rudolph, 2011; van Harmelen et al., 2007) incorporated first-order logical syntax with restrictions to ensure decidability and control tractability. A key feature of description logics is the separation of assertions from ontological knowledge. In description logics, the facts in a knowledge base, consisting of the attributes of entities and the relationships between them, are captured in the assertional box (ABox). Ontological knowledge about attributes and relationships are captured in the terminological box (TBox) and role box (RBox), respectively. The formal specification of valid ontological constraints in the TBox and RBox defined different classes of descrip-

tion logics. These specifications increased the clarity of the ramifications of supporting different types of ontological knowledge and enabled the construction of reasoning systems that demonstrated tractable performance on practically useful problems (Horrocks et al., 1999). Description logics also played a key role in the development of the Semantic Web (Baader et al., 2005).

2.3 Semantic Web: Ontologies and Tools

The Semantic Web movement was motivated by the prospect of exploiting the vast amount of information available on the Internet to build capable, knowledge-based system. In order to address the knowledge acquisition bottleneck, the Semantic Web movement proposed democratizing the task of specification and annotation (Berners-Lee et al., 2001; Antoniou and Van Harmelen, 2004). The key to this democratization was a set of standards for specifying knowledge using common representations and formats, and a common set of ontological conventions and primitives that could be easily adapted to many domains (Hitzler et al., 2009). The promise of these developments was a so-called "Semantic Web" of unambiguous, machine-readable information alongside the human-readable World Wide Web.

One of the key steps in realizing this promise was the definition of standard formats to express semantic knowledge (Decker et al., 2000). These formats include general purpose specifications, such as the Resource Description Format (RDF), RDF Schema, and the Web Ontology Language (Horrocks et al., 2003). These languages built on simple primitives such as subject-predicate-object triples. Many domain-specific specifications

have been built on top of these languages (Rector, 2003), such as the Friend-of-a-Friend (FOAF) (Golbeck and Rothstein, 2008) schema for social networks, the Functional Requirements of Bibliographic Records (FRBR) (Boeuf, 2001) for cataloging publications, creators, and subjects, the Music Ontology (Raimond et al., 2007), for musical works, and GALEN and SNOMED (Smith et al., 2005), for medical information.

The development of these standards seemed a precondition for addressing the problem of building systems to acquire knowledge. By allowing many different knowledge creators to share the same approach to representation, many hoped that semantically-annotated would become omnipresent among content on the World Wide Web. However, as the rate of content creation on the World Wide Web continued to accelerate, Semantic Web annotations of this data did not keep pace (Shadbolt et al., 2006).

2.4 Information and Knowledge Extraction

Concurrently with the development of tools and ontologies by the Semantic Web community, a number of advances were made in information extraction (Sarawagi, 2008). One of the keys to the rapid development in information extraction were improved techniques for natural language processing. Improvements in parsing (Klein and Manning, 2003; Collins and Koo, 2005; De Marneffe et al., 2006), part-of-speech tagging (Toutanova et al., 2003), semantic role labeling (Màrquez et al., 2008), and named entity recognition (Nadeau and Sekine, 2007; Collins and Singer, 1999), were the raw ingredients that enabled the information extraction advances of the last decade.

Building from these advances, many recent projects focus on extracting information

from text, including a number of efforts to extract structured knowledge. These projects usually adopted one of three models: (1) extracting information from structured textual inputs with a well-defined set of output targets; (2) extracting from unstructured text with no fixed set of output targets; or (3) extracting information from unstructured text while still maintaining a well-defined set of output targets. The first and third approaches are generally referred to as ontology-based information extraction (OBIE) (Wimalasuriya and Dou, 2010) while the second model is called open information extraction (OpenIE).

Projects that adopted the first model of information extraction often worked with information sources with regular and well-defined structure that were highly curated to maintain quality and minimize noise, with the online, user-constructed encyclopedia Wikipedia as a favorite example. Intelligence in Wikipedia (Weld et al., 2008), the DB-Pedia resource (Auer et al., 2007; Bizer et al., 2009) and YAGO (Suchanek et al., 2007, 2008; Kasneci et al., 2009) all use Wikipedia as input and produced knowledge bases that relate Wikipedia entities using structured labels, such as those found in informational tables on Wikipedia pages. These projects had a significant initial impact, since they quickly created a high-quality source of structured knowledge. However, a key limitation of these approaches was their need for high-quality, structured input, another instance of the knowledge acquisition bottleneck.

The second model of information extraction, OpenIE, took a drastically different approach to gathering knowledge (Etzioni et al., 2008; Christensen et al., 2011; Etzioni et al., 2011; Fader et al., 2011; Gamallo et al., 2012; Mausam et al., 2012). These systems relied heavily on natural language processing to analyze the structural features of naturally occurring text. Based on these features, OpenIE systems attempt to extract subjects,

objects, and determine the relationships between them in text. A key strength of this approach is generality: OpenIE can be applied to any text without defining target attributes or relationships in advance. However a major challenge for OpenIE is the quality and interpretability of the results. Errors in the underlying NLP systems can compromise the output of the information extraction. More problematically, the systems extract attributes and relationships, but determining whether these extractions have any semantically useful content is still an open challenge (Wang et al., 2011).

The third model of information extraction attempts to find a middle ground between these approaches by extracting knowledge that fits a known schema using naturally occurring text (as well as structured information, when available). Projects such as Elementary (Niu et al., 2012a), the Knowledge Vault (Dong et al., 2014a), NELL (Carlson et al., 2010a), PROSPERA (Nakashole et al., 2011), and StatSnowball (Zhu et al., 2009) all use, to varying extents, a well-defined set of entity attributes and relationships to extract. An attractive characteristic of these systems is that they are adaptable to a variety of textual corpora, but still produce readily interpretable results. One challenge facing these approaches is the curse of dimensionality: the textual input has a vast number of features, and in supervised settings training data is required to determine which features are reliable for extracting a given attribute or relationship.

One strategy for addressing this difficulty widely adopted by the information extraction community is the use of semi-supervised learning to apply a modest amount of training data to expand the scope of extractions. In particular, distant or weak supervision, where instead of training instances, training data takes the form of a high-precision rule or a pattern that can be applied to instances (Surdeanu et al., 2010; Nguyen and Mos-

chitti, 2011; Thomas et al., 2011; Krause et al., 2012; Takamatsu et al., 2012; Roth et al., 2013; Angeli et al., 2014; Fan et al., 2014). Usually a bootstrapping procedure is used to iteratively improve the model in these scenarios. While these techniques have fueled many different knowledge extraction efforts, many researchers have begun to use the outputs of these systems as the first stage in more sophisticated knowledge base construction systems using probabilistic modeling techniques.

2.5 Probabilistic Graphical Models and Structured Prediction

One of the key limitations of early approaches to reasoning in knowledge bases is the difficulty of capturing uncertainty, such as the uncertainty that arises from myriad extractions from text. Probabilistic models (Pearl, 1988) provide the mechanisms to capture both the uncertainty of the data. The advent of probabilistic graphical models (PGMs) has provided the machine learning community with a powerful way of expressing dependencies between random variables and building models to make predictions or infer missing information (Koller and Friedman, 2009). PGMs have proved particularly useful in cases when output is structured and a number of related judgments must be made, such as modeling entities, labels, and links in a network (Namata et al., 2011) or learning a mixture of sparse distributions describing discrete data (Blei et al., 2003).

A number of approaches have extended PGMs to allow models to be specified easily and include a richer set of features. Markov logical networks (Richardson and Domingos, 2006) and probabilistic soft logic (Broecheler et al., 2010) provide a first-order logic syntax that allows discrete and continuous Markov random fields, respectively, to be eas-

ily specified and lifted-inference (Braz et al., 2005) takes advantage of templated models during inference. Approaches such as conditional random fields (Lafferty et al., 2001), max-margin Markov networks (Taskar et al., 2003), SEARN (Daumé III et al., 2009) and SVM^{struct} (Tsochantaridis et al., 2004) combine the advantages of feature-rich classification approaches and the structural dependencies between outputs captured by PGMs.

2.6 Contemporary Approaches to Knowledge Base Construction

Many recent research projects seek to build knowledge bases from the noisy outputs of knowledge extraction systems (Nickel et al., 2015). These projects adopt diverse approaches to this task: some focus on fusing knowledge from different techniques to obtain a more robust knowledge base, others try to map the extractions into lower dimensional space to remove variability, and a third set of approaches uses the ontological constraints between extractions to develop measures of coherency.

Information extraction systems generate many extractions using differing techniques from diverse inputs. A number of approaches seek to exploit the agreements and disagreements in extractions to improve knowledge base construction. Dong et al. (2014b) describes knowledge fusion, a method for combining the outputs of multiple extraction techniques. Other approaches (Platanios et al., 2014; Balcan et al., 2013; Carlson et al., 2010b) use these patterns to estimate the error rate of extraction techniques, and, by extension, better understand the errors in the knowledge base.

A second set of techniques seeks to improve knowledge base construction by mapping the noisy knowledge in extractions to a lower dimensional space to reduce variability

and capture the essential statistical correlations between relationships and attributes in the knowledge base (Nickel et al., 2011, 2014; Yao et al., 2012, 2013; Socher et al., 2013). These approaches can often exploit popular models for dimensionality reduction through matrix factorization or representation learning with neural networks. This strength comes at a cost: the generality of these approaches makes incorporating the semantic relationships in knowledge bases difficult.

A key differentiator between "knowledge" and arbitrary data is the semantically meaningful relationships that exist between the extractions. A number of approaches attempt to capture these relationships by introducing constraints into the knowledge base construction process. PROSPERA (Nakashole et al., 2011) uses hard constraints derived from the YAGO ontology with a weighted MaxSAT reasoner. WebChild (Tandon et al., 2014b,a) encodes measures of coherency using an integer-linear program. Elementary (Niu et al., 2012b) and StatSnowball (Zhu et al., 2009) both include a collection of statistical models, including constraints specified with Markov logic to improve the consistency of the extracted knowledge base. Jiang et al. (2012) introduce a method based on Markov logic networks for all extractions to enforce ontological constraints on the extractions, discussed in more detail in subsequent chapters. ProPPR takes a different approach, by using random walks in a knowledge graph to determine the probability of extractions (Wang et al., 2014, 2015).

My work presents an alternate model for knowledge base construction that differs from the previous and contemporary work in a number of important ways. Systems such as PROSPERA and WebChild use specific, hand-coded constraints for their associated tasks and are difficult to optimize, whereas my work specifies a general model for the

common failure cases in knowledge graphs, allows a straightforward specification of the associated constraints, and implements scalable optimization for these constraints. StatSnowball and Elementary rely on simpler models such as logistic regression and conditional random fields, but these models do not incorporate the rich ontological domain knowledge and their results must be heuristically reconciled with the output of a Markov logic network. The work of Jiang et al. uses a Markov logic network and jointly optimizes over all extractions, but the choice of Markov logic severely limits the scalability of the approach. Finally, the idea of locally grounded models using random walks, such as ProPPR, can be seen as complementary to my work, as I use a partially grounded model for streaming inference. However, the model of knowledge graphs I propose captures the general ontological relationships in knowledge graphs and provides a cleaner way of integrating ontological information.

2.7 Streaming and Online Inference

Updating inference is a longstanding problem in artificial intelligence. The classic problem of belief revision (Gardenfors, 1992) considers revising and updating a set of propositional beliefs using a set of axiomatic guarantees to consistency. Diverse research has considered updating the parameters or structure of Bayesian networks in response to evolving evidence (e.g. Buntine, 1991; Friedman and Goldszmidt, 1997; Li et al., 2006). Finally, many models address dynamic or sequential data, such as Dynamic Bayesian Networks (Murphy, 2002) and hierarchical hidden Markov models (Fine et al., 1998). Our work addresses the specific problem of approximating full MAP inference in the online setting

when a model is given and provides formal guarantees for the approximation quality.

Making efficient updates to the full inference result is the goal of a related area of research, *adaptive inference*. Adaptive marginal inference (Acar et al., 2008; Sümer et al., 2011) can update the marginal probability of a query in $\mathrm{O}\left(2^{\mathrm{tw}(G)} \log n\right)$-time, where $\mathrm{tw}(G)$ is the tree-width of the graph and n is the number of variables. Adaptive MAP inference (Acar et al., 2009) can update the MAP state in $\mathrm{O}\left(m + m \log(n/m)\right)$-time, where m is the number of variables that change their state. Though the algorithm does not need to know m beforehand, a model change could result in changes to all n variables' states, with cost equivalent to exact inference. These adaptive inference techniques do not currently support partial updates to the MAP state or accommodate budgeted updates.

Approximate adaptive inference was considered by Nath and Domingos (2010), who proposed *expanding frontier belief propagation* (EFBP), a belief propagation algorithm that only updates messages in the vicinity of the updated potentials. They showed that the beliefs generated by EFBP lower- and upper-bound the beliefs of full BP, thereby providing guarantees on the quality of the approximation. This analysis differs from mine in that it bounds the individual marginal probabilities, whereas I bound the L^1 distance between MAP states. Unlike my approximation algorithm, EFBP does not explicitly limit computation and, in the worst case, may need to update all variables to achieve convergence conditions.

Another related line of work focuses on anytime inference. These methods perform belief propagation, either directly as in (Chechetka and Guestrin, 2010) or on using lifted inference as in (de Salvo Braz et al., 2009), with the goal of providing a bounded inference of query set given a time budget. This contrasts with my approach which is focused on

improving the overall MAP estimate in bounded time, rather than the estimate of a specific query set.

The quantity I call inference regret is conceptually similar to *collective stability* (London et al., 2013). Collective stability measures the amount of change in the output of a structured predictor induced by local perturbations of the evidence. London et al. (2013, 2014) analyzed the collective stability of marginal inference in discrete graphical models, concluding that (approximate) inference with a strongly convex entropy function enhances stability. Our technical approach is similar, in that it also leverages strong convexity. However, the types of perturbations I consider—fixing target variables—are not covered by their analysis. Stability analysis is closely related to *sensitivity analysis*. Since the terms are used interchangeably in the literature, I distinguish them as follows: sensitivity analysis examines *if* and *when* the solution changes; stability analysis examines *how much* it changes by. Laskey analyzed the sensitivity of queries (which can be used for marginal inference) in Bayesian networks. Chan and Darwiche studied the sensitivity of queries (2005) and MAP inference (2006) in Markov networks. Their 2005 paper also analyzes the stability of queries.

Chapter 3: Problem Formulation for Knowledge Graph Identification

Given the many different approaches and projects focused on knowledge base construction, choosing the appropriate problem setting and attendant formalisms is important. To provide a clearer perspective on the obstacles for large-scale knowledge base construction, we inventory common errors produced by information extraction systems, illustrated with anecdotes from a real-world system. By representing a knowledge base as a graph, we demonstrate how the prominent errors in knowledge graphs correspond to the fundamental tasks in an approach known as *graph identification*. One important ingredient for graph identification for knowledge graphs is ontological information. We devise a formal definition of knowledge graph identification, which combines candidates from information extraction and ontological information and show how this problem formulation addresses errors in knowledge graphs.

3.1 Knowledge Graphs

A *knowledge graph* is a structured knowledge base consisting of nodes and labeled, directed edges, $K = (\mathcal{V}_k, \mathcal{E}_k)$. Nodes in the knowledge graph correspond to entities. Each

node is associated with one or more of s indicator variables representing labels:

$$v \in \mathcal{V}_k \triangleq (l_1, l_2, \cdots l_s); \forall_{i \in (1 \cdots s)} l_i \in \{0, 1\}$$

These labels capture categorical class or type information and attributes of the entity. The edges of the knowledge graph are relationships between two nodes. Since there are many possible relationships in a knowledge graph, each edge is defined by the source and target nodes and a set of indicator variables representing possible relationships:

$$e \in \mathcal{E}_k \triangleq (v_{src}, v_{tgt}, r_1, r_2, \cdots r_t), r_i \in \{0, 1\}$$

The possible labels $(l_1, l_2, \cdots l_s)$ and relationships $(r_1, r_2, \cdots r_t)$ in the knowledge graph are defined by an ontology, also known as a *concept graph*. The dichotomy between the knowledge graph and the concept graph can be likened to the ABox and TBox in description logics (van Harmelen et al., 2007) or the data tables and schema in databases or RDF (Hitzler et al., 2009). However, the concept graph has a more structured form as a graph, where nodes are the possible labels and relationships in the knowledge graph, and edges between labels and relationships express ontological constraints. The many ontological relationships found in Semantic Web formats such as RDF and OWL are captured in the concept graph. Constraints such as domain (`rdf:domain`), range (`rdf:range`), label subsumption (`rdf:subClassOf`), relation subsumption (`rdf:subPropertyOf`), mutual exclusion of labels (`owl:disjointWith`), mutual exclusion of relations

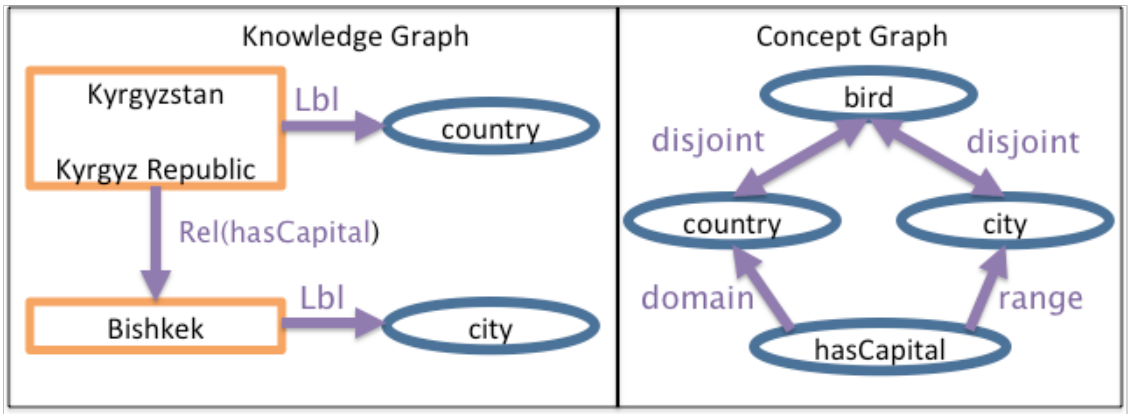


Figure 3.1: A simple example showing a knowledge graph, which captures entity labels and relationships between entities, and a concept graph, which captures ontological relationships between labels and relationships

(`owl:propertyDisjointWith`), inverse labels (`owl:complementOf`), and inverse relations (`owl:inverseOf`) can all be expressed as edges in the concept graph. For example, in a concept graph, mutually exclusive (`disjointWith`) labels are connected by an edge expressing the mutual exclusion relationship, while the label (class) that comprises the domain of a relation (property) can be specified by a domain edge between the class and the relation. Other ontological properties such as symmetry (`owl:SymmetricProperty`), functionality (`owl:FunctionalProperty`), transitivity (`owl:TransitiveProperty`), cardinality (`owl:cardinality`) can be specified as attributes of nodes in the concept graph. Ontological relationships with higher arities, e.g. intersection (`owl:intersectionOf`), and union (`owl:unionOf`), can be more difficult to express, requiring the inclusion of hyperedges in the graph. Figure 3.1 shows an example of a knowledge graph and its accompanying concept graph.

3.2 Common Errors in Information Extraction

Information extraction systems operate over many diverse datatypes, such as web pages, images and videos, and use a collection of strategies to generate candidate facts this data. Many of the popular and practically successful information extraction systems have focused on extracting knowledge from text. These systems use a host of techniques, spanning syntactic, lexical and structural features of text. Ultimately, these extraction systems produce candidate facts that include a set of entities, attributes of these entities, and the relations between these entities which can be viewed as a structure We call the *extraction graph*. However errors in the source data and the extraction process introduce inconsistencies in the extraction graph. Such noise obscures the true knowledge graph, which captures a consistent set of entities, attributes and relations. In this section, We inventory the differing errors made by information extraction systems and discuss their impact on the knowledge graph. We illustrate these errors and the accompanying challenges with examples taken from a real-world information extraction system, the Never-Ending Language Learner (NELL).

3.2.1 Entity Ambiguity

Entity recognition and mapping is one of the fundamental problems in information extraction. The task of recognizing entities in text is challenging for a number of reasons. Many textual analysis systems rely on parsing or part-of-speech tagging to identify entities, and both of these tasks suffer from errors. Often such systems fail to extract an entity, extract a spurious entity, or misidentify the boundary of the entity, producing an entity that includes

extraneous content or is not specific enough (Nadeau and Sekine, 2007). For example, NELL includes entities for “Obama” and “Barack Obama Obama”; the former entity is not specific enough to disambiguate from other members of the Obama family, while the latter may be the result of mistakenly concatenating an entity spanning two sentences.

Even when entity recognition systems operate correctly, the diversity of text on the WWW can introduce noise. Many textual references that initially look different may refer to the same real-world entity. For example, NELL’s knowledge base contains candidate facts involving the entities “kyrghyzstan”, “kyrgzstan”, “kyrgystan”, “kyrgyz republic”, “kyrgyzstan”, and “kyrgistan” which are all variants or misspellings of the country Kyrgyzstan found on the World Wide Web.

In the extraction graph, each of these incorrectly identified entities correspond to different nodes. In the correct knowledge graph, spurious nodes are removed and co-referent entities are collapsed into a single node. The task of *entity resolution* is used to determine co-referent entities in and produce a consistent set of labels and relations for each resolved node.

3.2.2 Attribute Errors

Another challenge in knowledge graph construction is inferring labels consistently. Information extraction systems use a variety of signals to predict labels, which range from extracting information using tables and links, to learning textual patterns indicative of a particular label. Since training data is scarce, the features associated with a particular label can be unreliable.

For example, NELL's extractions assign Kyrgyzstan the labels "country" as well as "bird." This error results from the interaction between an inconsistency in the source data and an inadequately robust feature in the extraction system. NELL extracted features from webpages on a birdwatching club website. These webpages would present catalogs of birds seen on birdwatching trips by the members of a birdwatching club. The vast majority of these webpages would contain hyperlinks where the displayed link text was the name of the bird, leading to a webpage with pictures of the named bird. However, in one instance a member posted a link to all birds sighted in a trip to the country Kyrgyzstan, and specified the link text as Kyrgyzstan, leading to an erroneous extraction.

Such errors are commonplace due to a mismatch between the numerous features available in large text corpora and the tiny amount of training data available to assess the relevancy of these features. While the extraction graph may contain many incorrect and inconsistent labels, the correct knowledge graph maintains a consistent set of labels. Ontological information suggests that an entity is very unlikely to be both a country and a bird at the same time, since these classes are mutually exclusive. Furthermore, Kyrgyzstan has a number of relationships with other entities, such as a `hasCapital` relationship with the entity Bishkek, which are inconsistent with the assertion that Kyrgyzstan is a bird. By using information from other labels and related facts, such inconsistencies can be removed from the knowledge graph. The task of determining the labels of a node by using features from neighboring nodes is called *collective classification*.

3.2.3 Relation Extraction Errors

A third problem commonly encountered in knowledge graphs is determining the relationships between entities. Similar to attribute prediction, information extraction systems use a host of features to determine whether a relationship exists between two extracted entities. Again, the staggering number of potential features is mismatched with the small amount of training data available for these applications, and errors are common.

For example, NELL also has many facts relating the location of Kyrgyzstan to other entities. These candidate relations include statements that Kyrgyzstan is located in Kazakhstan, Kyrgyzstan is located in Russia, Kyrgyzstan is located in the former Soviet Union, Kyrgyzstan is located in Asia, and that Kyrgyzstan is located in the US. Some of these possible relations are true, while others are clearly false and contradictory.

While the extraction graph may contain many spurious relationships, the correct knowledge graph includes a consistent set of links. Using the labels and other relationships of the nodes can aid the process of removing inconsistent links. The process of predicting edges between nodes is known as *link prediction*.

3.3 Graph Identification

In the previous section, we identified three core problems, entity ambiguity, attribute errors, and relation extraction errors, that appear in the extraction graph produced by an information extraction system. These three problems can be addressed by the tasks of entity resolution, collective classification, and link prediction, respectively. (Namata et al., 2011) refer to the process of inferring the structure of a graph through the combination of

entity resolution, collective classification, and link prediction as *graph identification*.

Graph identification relies on relational features, such as the labels and relations of neighboring nodes. One key observation about the graph identification problem settings is that the component tasks are intradependent as well as interdependent. For example, link prediction may be a function of the labels of neighboring nodes, but these labels are inferred by collective classification, which, in turn, depends on the correct resolution of entities and the links between them. As a result, graph identification benefits from a joint optimization, where entity resolution, collective classification, and link prediction are performed concurrently.

Graph identification can pose two substantial challenges: scalability and modeling. Jointly performing entity resolution, collective classification and link prediction is far more complex than completing each task separately. In particular, graph identification models produce output in a state space that is the product of the state space of each task – the results include all possible labels, all possible entity co-references, and all possible relationships between entities. Namata et al. sidesteps these problems with C^3, or coupled collective classifiers, where each of the three tasks is performed separately, using the outputs of the other two tasks as features in the prediction, with the process repeated until a convergence criterion is met. The second challenge in graph identification is model specification. While the tasks of entity resolution, collective classification, and link prediction are interrelated, determining the features that appropriately capture these interactions can be difficult.

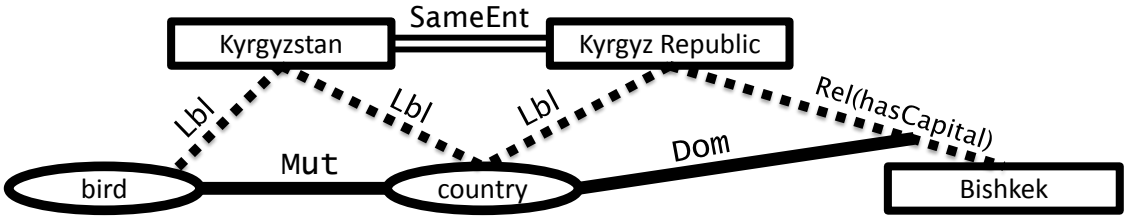


Figure 3.2: An illustration of the example showing how knowledge graph identification can resolve conflicting information in an extraction graph. Entities are shown in rectangles, dotted lines represent uncertain information, solid lines show ontological constraints and double lines represent co-referent entities found with entity resolution.

3.4 Adapting Graph Identification to Knowledge Graphs

Refining an extraction graph to produce a knowledge graph is an instance of graph identification. Features from an information extraction system, such as confidence scores assigned to candidate extractions, can provide the basic features for graph identification. However knowledge graphs have a key ingredient that differentiates them from the general graph identification setting: the ontological knowledge found in the concept graph. The concept graph provides a rich source of relational information that mediates dependencies between the facts in the knowledge graph. These dependencies allow a sophisticated approach to resolving the knowledge graph, while providing a well-founded specification for modeling interdependencies between the tasks in graph identification.

Figure 3.2 illustrates a complex example of how ontological information can be used to resolve a knowledge graph. As mentioned earlier, NELL's ontology includes the constraint that the labels "bird" and "country" are mutually exclusive. Reasoning collectively allows us to resolve which of these two labels is more likely to apply to Kyrgyzstan. For example, NELL is highly confident that the Kyrgyz Republic has a capital

city, Bishkek. The NELL ontology specifies that the domain of the relation "hasCapital" has label "country." Entity resolution allows us to infer that "Kyrgyz Republic" refers to the same entity as "Kyrgyzstan." Deciding whether Kyrgyzstan is a bird or a country now involves a prediction which includes the confidence values of the corresponding "bird" and "country" facts from co-referent entities, as well as collective features from ontological relationships of these co-referent entities, such as the confidence values of the "hasCapital" relations.

We refer to the process of inferring a knowledge graph from a noisy extraction graph using statistical features and ontological constraints as knowledge graph identification. The central hypothesis on this dissertation is that the collective dependencies found in a knowledge graph are paramount to the goal of knowledge graph construction. The problem definition of knowledge graph identification admits many possible models capable of meeting the requirements of the problem definition. In the next chapter, we define a probabilistic graphical model for knowledge graph identification that meets these requirements and tests the hypothesis that knowledge graph identification will improve the quality of the knowledge graph.

Chapter 4: Modeling Knowledge Graph Identification

The probabilistic model we define for knowledge graph identification uses a modeling framework called probabilistic soft logic (PSL). We provide background on PSL in the next section, and then use PSL to define a model for knowledge graph identification. This model captures both the uncertainty from information extraction systems and the key constraints between facts found in the ontology. Next, we convert this model of knowledge graph identification into a probability distribution over the space of possible knowledge graphs, and use this distribution to determine the most probable knowledge graph. Finally, we show the efficacy of this approach in a number of empirical experiments.

4.1 Background: PSL for Knowledge Graphs

Probabilistic soft logic (PSL) (Bach et al., 2015; Broecheler et al., 2010; Kimmig et al., 2012) is a recently-introduced framework which allows users to specify rich probabilistic models over continuous-valued random variables. Like other statistical relational learning languages such as Markov Logic Networks (MLNs), it uses first-order logic to describe features that define a Markov network. In contrast to other approaches, PSL employs continuous-valued random variables rather than binary variables and casts most probable explanation (MPE) inference as a convex optimization problem that is significantly more

efficient to solve than its combinatorial counterpoint (polynomial vs. exponential).

A PSL model is composed of a set of weighted, disjunctive first-order logic rules, where each rule defines a set of features of a Markov network sharing the same weight. Consider the formula

$$\mathrm{P}(A, B) \wedge \mathrm{Q}(B, C) \overset{w}{\Rightarrow} \mathrm{R}(A, B, C)$$

which is an example of a PSL rule. Here w is the weight of the rule, A, B, and C are universally-quantified variables, and P, Q and R are predicates. A *grounding* of a rule comes from substituting constants for universally-quantified variables in the rule's atoms. In this example, assigning constant values $A = \mathtt{a}$, $B = \mathtt{b}$, and $C = \mathtt{c}$ to the variables in the rule above would produce the ground atoms $\mathrm{P}(\mathtt{a}, \mathtt{b})$, $\mathrm{Q}(\mathtt{b}, \mathtt{c})$, $\mathrm{R}(\mathtt{a}, \mathtt{b}, \mathtt{c})$. Each ground atom takes a soft-truth value in the range $[0, 1]$.

PSL associates a numeric *distance to satisfaction* with each ground rule that determines the value of the corresponding feature in the Markov network. The distance to satisfaction is defined by treating the ground rule as a formula over the ground atoms in the rule. In particular, PSL uses the *Lukasiewicz t-norm* and *co-norm* to provide a relaxation of the logical connectives, AND ($\wedge$), OR($\vee$), and NOT($\neg$), as follows:

$$p \wedge q = \max(0, p + q - 1) \tag{4.1}$$

$$p \vee q = \min(1, p + q) \tag{4.2}$$

$$\neg p = 1 - p \tag{4.3}$$

This relaxation coincides with Boolean logic when p and q are in $\{0, 1\}$, and provides a consistent interpretation of soft-truth values when p and q are in the numeric range $[0, 1]$.

A PSL program, $\mathbf{\Pi}$, consisting of a model as defined above, along with a knowledge base of atoms, F, produces a set of ground rules, R by substituting the universally quantified atoms in a rule with ground atoms from F . If I is an interpretation (an assignment of soft-truth values to ground atoms) and r is a ground instance of a rule, then the distance to satisfaction $\phi_r(I)$ of r is $1 - T_r(I)$, where $T_r(I)$ is the soft-truth value from the Lukasiewicz t-norm. This distance to satisfaction takes the form of a hinge-function, a commonly-used, convex relaxation of 0-1 loss that has many tractability benefits. For this reason the class of probabilistic models implemented by PSL are known as *hinge-loss Markov random fields* (HL-MRFs).

We can define a probability distribution over interpretations by combining the weighted degree of satisfaction over all ground rules, R, and normalizing, as follows:

$$f(I) = \frac{1}{Z} \text{exp} \left[- \sum_{r \in R} w_r \phi_r(I)^p \right] \tag{4.4}$$

Here Z is a normalization constant, w_r is the weight of rule r, and p in $\{1, 2\}$ allows a linear or quadratic combination of rules. Thus, a PSL program (set of weighted rules and facts) defines a probability distribution from a logical formulation that expresses the relationships between random variables.

MPE inference in PSL determines the most likely soft-truth values of unknown ground atoms using the values of known ground atoms and the dependencies between atoms encoded by the rules, corresponding to inference of random variables in the under-

lying Markov network. PSL atoms take soft-truth values in the interval $[0, 1]$, in contrast to MLNs, where atoms take Boolean values. MPE inference in MLNs requires optimizing over combinatorial assignments of Boolean truth values. In contrast, the relaxation to the continuous domain greatly changes the tractability of computations in PSL: finding the most probable interpretation given a set of weighted rules is equivalent to solving a convex optimization problem. Recent work from Bach et al. (2012) introduces a consensus optimization method applicable to PSL models; their results suggest consensus optimization scales linearly with the number of ground rules in the model.

4.2 Model for Knowledge Graph Identification

Knowledge graphs contain three types of facts: facts about entities, facts about entity labels and facts about relations. We represent entities with the logical predicate ENT(E) and labels with the logical predicate LBL(E,L) where entity E has label L. Relations are represented with the logical predicate REL(E_1,E_2,R) where the relation R holds between the entities E_1 and E_2, eg. `R(E1,E2)`.

In knowledge graph identification, our goal is to identify a true set of atoms from a set of noisy extractions. Our method for knowledge graph identification incorporates three components: capturing uncertain extractions, performing entity resolution, and enforcing ontological constraints. We show how to create a PSL program that encompasses these three components, and then relate this PSL program to a distribution over possible knowledge graphs.

4.2.1 Representing Uncertain Extractions

We relate the noisy extractions from an information extraction system to the above logical predicates by introducing *candidate* predicates, using a formulation similar to Jiang et al. (2012). For each candidate entity, we introduce a corresponding predicate, CANDENT(E). Labels or relations generated by the information extraction system correspond to predicates CANDLBL(E,L) or CANDREL(E_1,E_2,R) in the system. Uncertainty in these extractions is captured by assigning these predicates a soft-truth value equal to the confidence value from the extractor. For example, the extraction system might generate a relation, `hasCapital(kyrgyzstan, Bishkek)` with a confidence of .9, which we would represent as CANDREL(`kyrgyzstan`,`Bishkek`, `hasCapital`) and assign it a truth value of .9.

Information extraction systems commonly use many different extraction techniques to generate candidates. For example, NELL produces separate extractions from lexical, structural, and morphological patterns, among others. We represent metadata about the technique used to extract a candidate by using separate predicates for each technique T, of the form CANDREL_T and CANDLBL_T. These predicates are related to the true values of attributes and relations we seek to infer using weighted rules.

$$\text{CANDREL}_T(E_1, E_2, R) \Rightarrow \text{REL}(E_1, E_2, R) \tag{4.5}$$

$$\text{CANDLBL}_T(E, L) \Rightarrow \text{LBL}(E, L) \tag{4.6}$$

Together, we denote the set of candidates, generated from grounding the rules above using

the output from the extraction system, as the set $\mathcal{C}$.

4.2.2 Entity Resolution

While the previous PSL rules provide the building blocks of predicting links and labels using uncertain information, knowledge graph identification employs entity resolution to pool information across co-referent entities. A key component of this process is identifying possibly co-referent entities and determining the similarity of these entities, which we discuss in detail in Section 4.4. We use the SAMEAS predicate to capture the similarity of two entities, for example SAMEAS(`kyrgyzstan`, `kyrgz republic`).

To perform entity resolution using the SAMEAS predicate we introduce three rules, whose groundings we refer to as $\mathcal{S}$, to our PSL program:

$$\textsc{SameAs}(E_1, E_2) \wedge \textsc{Lbl}(E_1, L) \Rightarrow \textsc{Lbl}(E_2, L) \tag{4.7}$$

$$\textsc{SameAs}(E_1, E_2) \wedge \textsc{Rel}(E_1, E, R) \Rightarrow \textsc{Rel}(E_2, E, R) \tag{4.8}$$

$$\textsc{SameAs}(E_1, E_2) \wedge \textsc{Rel}(E, E_1, R) \Rightarrow \textsc{Rel}(E, E_2, R) \tag{4.9}$$

These rules define an equivalence class of entities, such that all entities related by the SAMEAS predicate must have the same labels and relations. The soft-truth value of the SAMEAS, derived from our similarity function, mediates the strength of these rules. When two entities are very similar, they will have a high truth value for SAMEAS, so any label assigned to the first entity will also be assigned to the second entity. On the other hand, if the similarity score for two entities is low, the truth values of their respective

labels and relations will not be strongly constrained. We introduce these rules as weighted rules in the PSL model, where the weights can capture the reliability of the similarity function.

4.2.3 Enforcing Ontological Constraints

In our PSL program we also leverage rules corresponding to an ontology, the groundings of which are denoted as $\mathcal{O}$. These ontological rules are based on the logical formulation proposed in Jiang et al. (2012). Each type of ontological relation is represented as a predicate, and these predicates represent ontological knowledge of the relationships between labels and relations. For example, the ontological predicates DOM(`hasCapital`, `country`) and RNG(`hasCapital`, `city`) specify that the relation `hasCapital` is a mapping from entities with label `country` to entities with label `city`. The predicate MUT(`country`, `city`) specifies that the labels `country` and `city` are mutually exclusive, so that an entity cannot have both the labels `country` and `city`. We similarly use predicates for subsumption of labels (SUB) and relations(RSUB), and inversely-related functions (INV). To use this ontological knowledge, we introduce rules relating each ontological predicate to the predicates representing our knowledge graph. We specify seven types of ontological constraints in our experiments using weighted rules:

$$\textsc{Dom}(R, L) \quad \wedge \textsc{Rel}(E_1, E_2, R) \quad \Rightarrow \textsc{Lbl}(E_1, L) \tag{4.10}$$

$$\textsc{Rng}(R, L) \quad \wedge \textsc{Rel}(E_1, E_2, R) \quad \Rightarrow \textsc{Lbl}(E_2, L) \tag{4.11}$$

$$\textsc{Inv}(R, S) \quad \wedge \textsc{Rel}(E_1, E_2, R) \quad \Rightarrow \textsc{Rel}(E_2, E_1, S) \tag{4.12}$$

$$\textsc{Sub}(L, P) \quad \wedge \textsc{Lbl}(E, L) \quad \Rightarrow \textsc{Lbl}(E, P) \tag{4.13}$$

$$\textsc{RSub}(R, S) \quad \wedge \textsc{Rel}(E_1, E_2, R) \quad \Rightarrow \textsc{Rel}(E_1, E_2, S) \tag{4.14}$$

$$\textsc{Mut}(L_1, L_2) \quad \wedge \textsc{Lbl}(E, L_1) \quad \Rightarrow \neg\textsc{Lbl}(E, L_2) \tag{4.15}$$

$$\textsc{RMut}(R, S) \quad \wedge \textsc{Rel}(E_1, E_2, R) \quad \Rightarrow \neg\textsc{Rel}(E_1, E_2, S) \tag{4.16}$$

4.3 Implementing Knowledge Graph Identification with PSL

Combining the logical rules introduced in this chapter with atoms, such as candidates from the information extraction system (e.g. CANDREL(`kyrgyzstan`, `Bishkek`, `hasCapital`)), co-reference information from an entity resolution system (e.g. SAMEAS(`kyrgyzstan`, `kyrgz republic`)) and ontological information (e.g. DOM(`hasCapital`, `country`)) we can define a PSL program, $\mathbf{\Pi}$. The inputs to this program instantiate a set of ground rules, R, that consists of the union of groundings from uncertain candidates, $\mathcal{C}$, co-referent entities, $\mathcal{S}$, and ontological relationships, $\mathcal{O}$. The distribution over interpretations, I, generated by PSL corresponds to a probability

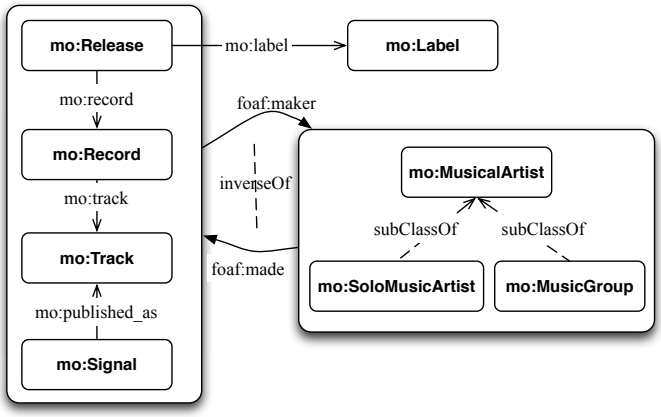


Figure 4.1: Subset of Music Ontology mapped using LinkedBrainz for MusicBrainz data in our synthetic dataset

distribution over knowledge graphs, G:

$$P_{\mathbf{\Pi}}(G) = f(I) = \frac{1}{Z}\exp\left[\sum_{r \in R} w_r \phi_r(I)^p\right] \tag{4.17}$$

The results of inference provide us with the most likely interpretation, or soft-truth assignments to entities, labels and relations that comprise the knowledge graph. By choosing a threshold on the soft-truth values in the interpretation, we can select a high-precision set of facts to construct a knowledge graph. Appendix A provides a sample PSL program that demonstrates how all of these components are implemented in code.

4.4 Experimental Evaluation

4.4.1 Datasets and Experimental Setup

We evaluate our method on two different datasets: a synthetic knowledge base derived from the LinkedBrainz project (Dixon and Jacobson), which maps data from the Mu-

sicBrainz community using ontological information from the MusicOntology (Raimond et al., 2007) as well as web-extraction data from the Never-Ending Language Learning (NELL) project (Carlson et al., 2010a). Our goal is to assess the utility of knowledge graph identification, formulated as a PSL model, at inferring a knowledge graph from noisy data. Additionally, we contrast two very different evaluation settings. In the first, as used in previous work Jiang et al. (2012) inference is limited to a subset of the knowledge graph generated from the test or query set. In the second evaluation setting, inference produces a complete knowledge graph, which is not restricted by the test set but employs a soft-truth threshold for atoms. We provide documentation, code and datasets to replicate our results on GitHub.[1]

MusicBrainz

MusicBrainz is a community-driven, open-source, structured database for music metadata, including information about artists, albums, and tracks, The Music Ontology is built on top of many well known ontologies, such as FRBR (Davis et al., 2005) and FOAF (Brickley and Miller, 2010), and has been used widely, for instance in BBC Music Linked Data sites (Kobilarov et al., 2009). However, the relational data available from MusicBrainz are expressed in a proprietary schema that does not map directly to the Music Ontology. To bridge this gap, the LinkedBrainz project publishes an RDF mapping between the freely available MusicBrainz data and the Music Ontology using D2RQ (Bizer and Seaborne, 2004). A summary of the labels and relations we use in our data is shown in Figure 4.1. We use an intuitive mapping of ontological relationships to the PSL pred-

[1] `https://github.com/linqs/KnowledgeGraphIdentification`

icates, using ontological information from FRBR and FOAF classes used by the Music Ontology. Specifically we convert `rdfs:domain` to DOM, `rdfs:range` to RNG, `rdfs:subClassOf` to SUB, `rdfs:subPropertyOf` to RSUB, `owl:inverseOf` to INV, and `owl:disjointWith` to MUT.

Our synthetic knowledge graph uses a sample of data from the LinkedBrainz mapping of the MusicBrainz project[2] and adds noise to generate a realistic data set. To generate a subset of the LinkedBrainz data, we use snowball sampling from a set of tracks in the MusicBrainz dataset to produce a set of recordings, releases, artists and labels. Next, we introduce noise into this graph by randomly removing known facts and adding inconsistent facts as well as generating random confidence values for these facts. This noise can be interpreted as errors introduced by a MusicBrainz user misspelling artist names, accidentally switching input fields, or omitting information when contributing to the knowledge base.

We model these errors by distorting a percentage of the true input data. For labels, we omit known labels and introduce spurious labels for 25% of the facts in the input data. When dealing with relations, We focus on the `foaf:maker` and `foaf:made` relations between artists and creative works. We randomly remove one of these pair of relations 25% of the time. Finally, 25% of the time we remove the relationship between a work and its artist, and insert a new relationship between the work and a generated artist, adding a SAMEAS for these two artists. The confidence values for facts found in the input are generated from a Normal$(.7, .2)$ distribution while inconsistent facts have lower confidence values generated from a Normal$(.3, .2)$ distribution. The high variance in these

[2] `http://linkedbrainz.c4dmpresents.org/content/rdf-dump`

distributions ensures a significant overlap. For the SAMEAS the similarity values are generated from a Normal$(.9, .1)$ distribution. In all cases, the distribution is thresholded to the $[0, 1]$ range. We summarize important data statistics in Table 4.1.

NELL

The goal of NELL is to iteratively generate a knowledge base. In each iteration, NELL uses facts learned from the previous iteration and a corpus of web pages to generate a new set of candidate facts. NELL selectively promotes those candidates that have a high confidence from the extractors and obey ontological constraints with the existing knowledge base to build a high-precision knowledge base. We present experimental results on the 165th iteration of NELL, using the candidate facts, promoted facts and ontological relationships that NELL used during that iteration. We summarize the important statistics of this dataset in Table 4.1. Due to the diversity of the web, the data from NELL is larger, includes more types of relations and categories, and has more ontological relationships than our synthetic data.

NELL uses diverse extraction sources, and in our experiments we use distinct predicates CANDLBL$_T$ and CANDREL$_T$ for the NELL sources CBL (Constraint Based Learner), CMC (Coupled Morphological Classifier), CPL (Coupled Pattern Learner), Morph (Morphological Patterns), and SEAL (Structured Extraction) while the remaining sources, which do not contribute a significant number of facts, are represented with CANDLBL and CANDREL predicates. In addition to candidate facts, NELL uses a heuristic formula to "promote" candidates in each iteration of the system into a knowledge base,

however these promotions are often noisy so the system assigns each promotion a confidence value. We represent these promoted candidates from previous iterations as an additional source with corresponding candidate predicates.

In addition to data from NELL, we use data from the YAGO database (Suchanek et al., 2007) as part of our entity resolution approach. Our model uses a SAMEAS predicate to capture the similarity of two entities. To correct against the multitude of variant spellings found in the data, we use a mapping technique from NELL's entities to Wikipedia articles, described in more detail below. We then define a similarity function on the article identifiers, which in this case are URLs, using the similarity as the soft-truth value of the SAMEAS predicate.

The YAGO database contains entities which correspond to Wikipedia articles, variant spellings and abbreviations of these entities, and associated WordNet categories. Our approach to entity resolution matches entity names in NELL with YAGO entities. We perform selective stemming on the NELL entities, employ blocking on candidate labels, and use a case-insensitive string match to find corresponding YAGO entities. Once we find a matching set of YAGO entities, we can generate a set of Wikipedia URLs that map to the corresponding NELL entities. We can judge the similarity of two entities by computing a set-similarity measure on the Wikipedia URLs associated with the entities. For our similarity score we use the Jaccard index, the ratio of the size of the set intersection and the size of the set union.

Table 4.1: Summary of dataset statistics for NELL and MusicBrainz, including (a) the number of candidate facts in input data, the distinct relations and labels present, and (b) the number of ontological relationships defined between these relations and labels

(a)

	NELL	MusicBrainz
Cand. Label	1.2M	320K
Cand. Rel	100K	490K
Promotions	440K	0
Unique Labels	235	19
Unique Rels	221	8

(b)

	NELL	MusicBrainz
DOM	418	8
RNG	418	8
INV	418	2
MUT	17.4K	8
RMUT	48.5K	0
SUB	288	21
RSUB	461	2

4.4.2 Learning Model Weights from Training Data

Our PSL model for knowledge graph identification defines a number of rules with weights that capture the importance of the rule in the probability distribution over knowledge graphs. While these weights can be manually specified, a more powerful approach is to learn the value of these weights from training data. The general methods for weight learning in PSL were introduced in (Bach et al., 2013), however the knowledge graph setting introduces some new considerations.

A complication of learning from training data is that, given the vast number of potential facts in knowledge graph, obtaining labeled data with sufficient coverage over the

facts in the knowledge graph can be difficult. Since most knowledge graph construction tasks are focused on a single, large dataset, weight learning takes place in a transductive setting and it can be difficult to separate a training network from the knowledge graph. Yet another problem is sparsity in knowledge graphs, which creates skew in the learning problem due to the large number of negative labels for false relationships and labels, which overwhelm the number of true labels and relationships.

We deal with the challenges of weight learning by carefully defining a training setup that efficiently uses training data by adopting a semi-supervised approach to learning. We define a training network by creating a ground graphical model using the facts in the training set. In addition to these facts, we approximate the contextual knowledge of the training data with the 2-hop neighborhood of the training variables, which we refer to as the training network.

Since the collective KGI model will include facts that are not part of the training network, we modify the model to scope the rules so that only those rules pertaining to the facts in the training network are grounded, and additional variables are not inferred. Next, we extend the training data to achieve greater coverage by performing inference for the training network using initial model weights while conditioning on the training set. This inference assigns values to the unknown variables in the 2-hop neighborhood of the training data. This inferred knowledge graph constitutes the labels for weight learning.

We learn weights using the maximum likelihood estimation approach implemented in PSL as reported in Bach et al. (2013). The MLE weight learning attempts to optimize weights to recover the full training network (both the training set and the Markov blanket) using the candidate facts and a small set of seed observations. While this approach could

be extended by using an iterative estimation-maximization algorithm that repeatedly infers the training network and the optimizes the model weights, we found a straightforward approach reduces training time while maintaining the weight quality.

4.4.3 Open-World vs Closed-World Evaluation Setting

In our experiments using NELL, we consider two scenarios. The first setting is based on a network similar to the training network used for weight learning discussed in the previous subsection. This setting is based on previous work by Jiang et al. (2012), and was defined to improve scalability by generating a grounding of the graphical model based on the test set, determining a 2-hop neighborhood of the test set, and then only including atoms that are not trivially satisfied in this grounding. In practice, this produces a neighborhood that is distorted by omitting atoms that may contradict those in the test set.

For example, if ontological relationships such as RNG(`hasCapital`,`country`) and MUT(`country`, `city`) are present, the test set atom LBL(`kyrgyzstan`,`country`) would not introduce LBL(`kyrgyzstan`,`city`) or REL(`kyrgyzstan`, `Bishkek`, `hasCapital`) into the neighborhood, even if supportive or contradictory data were present in the input candidates. By removing the ability to reason about supportive and contradictory information, we believe this evaluation setting diminishes the true difficulty of the problem. We validate our approach on this setting, but also present results from a more realistic setting.

In the second scenario we perform inference independently of the test set, lazily generating truth values for atoms supported by evidence. This lazy inference approach

performs repeated inferences. Each iteration uses the available evidence and previous inferences and grounds out the model, treating any inference below a soft-truth threshold of .01 as false (PSL's default behavior). In the context of the previous example, initially the test fact LBL(kyrgyzstan,country) would only be supported by candidate extractions for that particular fact. However, in subsequent iterations the inferred value of LBL(kyrgyzstan,country) would be influenced by other candidates not found in the test set. For example, a contradictory candidate CANDLBL(kyrgyzstan,city) could diminish this truth value, while a candidate REL(kyrgyzstan, Bishkek, hasCapital) ould increase the truth value. This iterative process of inference and grounding is repeated until no new groundings are added to the model. This second setting allows us to infer a complete knowledge graph similar to the MusicBrainz setting.

4.4.4 Results for Closed-World Settings

MusicBrainz

Our experiments on MusicBrainz data attempt to recover the knowledge graph despite the addition of noise which introduces uncertainty for facts, removes true information and adds spurious labels and relations. The synthetic modifications allow us to retain information about the true, complete knowledge graph and thus enumerate the full set of target facts. We evaluate the results by comparing to the true knowledge graph used to generate the data, and include false labels corresponding to spurious data introduced.

In our experiments, we represent the noisy relations and labels of the knowledge graph as candidate facts in PSL with the predicates CANDLBL and CANDREL. Our exper-

Table 4.2: A comparison of knowledge graph identification methods on MusicOntology data shows knowledge graph identification effectively combines the strengths of graph identification and reasoning with ontological information and produces superior results.

Method	AUC	Prec	Recall	F10.5	F10.0
Baseline	0.672	0.946	0.477	0.634	0.788
PSL-EROnly	0.797	0.953	0.558	0.703	0.831
PSL-OntOnly	0.753	0.964	0.605	0.743	0.832
PSL-KGI-Complete	**0.901**	**0.970**	**0.714**	**0.823**	**0.919**

iments use quadratic potentials, and static weights for all rules, where $w_{CL} = w_{CR} = 1$, $w_{EL} = w_{ER} = 25$ and $w_O = 100$. We evaluate a number of variants of the KGI model on their ability to recover this knowledge graph. We measure performance using a number of metrics: the area under the precision-recall curve (AUC), as well as the precision, recall and F1 score at a soft-truth threshold of $.5$ (the intuitive boundary between true and false in soft logic). Due to the high variance of confidence values and large number of true facts in the ground truth, the maximum F1 value occurs at a soft-truth threshold of 0, where recall is maximized, for all variants, and we report these results for completeness. These results are summarized in Table 4.2.

The first variant we consider uses only the input data, setting the soft-truth value equal to the generated confidence value as an indicator of the underlying noise in the data. The baseline results use only the candidate rules we introduced in subsection 4.2.1. We improve upon this data by adding either the entity resolution rules introduced in subsection 4.2.2, which we report as PSL-EROnly, or with weighted rules capturing ontological constraints introduced in subsection 4.2.3. Finally, we combine all the elements of knowledge graph identification introduced in Section 4.2 and report these results as PSL-KGI-Complete.

Table 4.3: Comparing against previous work on the NELL dataset, knowledge graph identification using PSL demonstrates a substantive improvement.

Method	AUC	Prec	Recall	F1
Baseline	0.873	0.781	0.881	0.828
NELL	0.765	0.801	0.580	0.673
MLN	0.899	**0.837**	0.837	0.836
PSL-KGI	**0.904**	0.767	**0.956**	**0.851**

The results on the baseline demonstrate the magnitude of noise in the input data; less than half the facts in the knowledge graph can be correctly inferred. Reasoning jointly about co-referent entities, as in graph identification, improves results. Using ontological constraints, as previous work in improving extraction in this domain has, also improves results as well. Comparing these two improvements, adding entity resolution has a higher AUC, while ontological constraints show a greater improvement in F1 score. However, when these two approaches are combined, as they are in knowledge graph identification, results improve dramatically. Knowledge graph identification increases AUC, precision, recall and F1 substantially over the the other variants, improving AUC and F1 over 10% compared to the more competitive baseline methods. Overall, we are able to infer 71.4% of true relations while maintaining a precision of $.97$. Moreover, a high AUC of $.901$ suggests that knowledge graph identification balances precision and recall for a wide range of parameter values.

NELL

While results on data with synthetic noise confirm our hypothesis, we are particularly interested in the results on a large, noisy real-world dataset. We apply knowledge graph

identification to data from iteration 165 of NELL, a dataset that has been previously studied and for which has a large, manually-labeled evaluation set has been collected (Jiang et al., 2012). We compare KGI to previous work, and a summary of results is shown in Table 4.3. Additional results on this dataset can be found in Appendix B

The first method we compare to is a baseline similar to the one used in the MusicBrainz results where candidates are given a soft-truth value equal to the extractor confidence (averaged across extractors when appropriate). Since this model is untrained, we report results at a soft-truth threshold of .45 which maximizes F1 and provides the most competitive baseline performance.

We also compare the default strategy used by the NELL project to choose candidate facts to include in the knowledge base. Their method uses the ontology to check the consistency of each proposed candidate with previously promoted facts already in the knowledge base. Candidates that do not contradict previous knowledge are ranked using a heuristic rule based on the confidence scores of the extractors that proposed the fact, and the top candidates are chosen for promotion subject to score and rank thresholds. Note that the NELL method includes judgments for all input facts, not just those in the test set.

The third method we compare against is the best-performing MLN model from Jiang et al. (2012), that expresses ontological constraints, and candidate and promoted facts through logical rules similar to those in our model. The MLN uses additional predicates that have confidence values taken from a logistic regression classifier trained using manually labeled data. The MLN uses hard ontological constraints, learns rule weights considering rules independently and using logistic regression, scales weights by the extractor confidences, and uses MC-Sat with a restricted set of atoms to perform approx-

imate inference, reporting output at a 0.5 marginal probability cutoff, which maximizes the F1 score. The MLN method only generates predictions for a 2-hop neighborhood generated by conditioning on the values of the query set, as described earlier.

Our method, PSL-KGI, uses PSL with quadratic, weighted rules for ontological constraints, entity resolution, and candidate and promoted facts as well as incorporating a prior. We also incorporate the predicates generated for the MLN method for a more equal comparison. We learn weights for all rules, including the prior, using a voted perceptron learning method. The weight learning method generates a set of target values by running inference and conditioning on the training data, and then chooses weights that maximize the agreement with these targets in absence of training data. Since we represent extractor confidence values as soft-truth values, we do not scale the weights of these rules. Using the learned weights, we perform inference on the same neighborhood defined by the query set that is used by the MLN method. We report these results, using the standard soft-truth threshold of 0.5, as PSL-KGI. As Table 4.3 shows, knowledge graph identification produces modest improvements in both F1 and AUC.

4.4.5 Results for Model Ablation Study

To better understand the contributions of various components of our model, we explore variants that omit one aspect of the knowledge graph identification model. In each case, we learn new weights for the remaining rules, and perform inference in the closed-world setting, and then measure performance on the test set. The results of the ablation study are shown in Table 4.4, while the precision-recall tradeoff for each method is shown in

Table 4.4: Comparing variants of PSL graph identification show the importance of ontological information, but the best performance is achieved when all of the components of knowledge graph identification are combined.

Method	AUC	Prec	Recall	F1
Baseline	0.873	0.781	0.881	0.828
PSL-NoSrcs	0.900	0.770	**0.955**	0.852
PSL-NoER	0.899	0.778	0.944	**0.853**
PSL-NoOnto	0.887	**0.813**	0.839	0.826
PSL-KGI	**0.904**	0.777	0.944	**0.853**

Fig. 4.2, and detailed results for each ablated method can be found in Appendix B.

The Baseline method uses the same methodology in the previous section, and does not include any collective knowledge graph identification rules. PSL-NoSrcs removes predicates CANDLBL$_T$ and CANDREL$_T$ for different candidate sources, replacing them with a single CANDLBL or CANDREL with the average confidence value across sources. PSL-NoER removes rules from subsection 4.2.2 used to reason about co-referent entities. PSL-NoOnto removes rules from subsection 4.2.3 that use ontological relationships to constrain the knowledge graph. We compare these methods to the full knowledge graph identification model, PSL-KGI.

The Baseline method had the worst performance, since it relies exclusively on extractor confidences to construct the knowledge graph. Examining the results, we find that the facts where the Baseline method has the greatest disagreement with our KGI model are those requiring ontological knowledge. For example, the Baseline method infers the `teamWonTrophy` for the "trophy" `playoffs` for many teams (e.g. `cubs`, `packers`, `colts`). While it may be true these teams won the playoffs, playoffs do not have label `trophy`, and so these facts are not germane for the relation.

Removing source information from the model had a minor impact on model performance, resulting in lower precision, higher recall, and lower F1 and AUC. Examining the most salient differences in the inferred knowledge bases, the major difference is that the full KGI model infers much higher truth values for a number of labels. For example, the full KGI model infers that `twilight`, `sideways` and `doctor_zhivago` have labels `movie` and `creativework` while the NoSrcs model has much lower truth values for these facts. This may be the result of information loss when a specialized, high-precision extractor for a particular domain is merged with lower precision extractors.

When the rules for entity resolution are removed from the model, the results mirror those of the full KGI model with a minor increase in precision and a minor decrease in AUC. Sampling prominent differences from the results, we find that the KGI model that includes entity resolution is able to extend inferences to aliases and variant names for the same entity. For example, the full KGI model determines that `ampalaya`, more frequently referred to as ”bitter melon”, is a food and vegetable. Another example is that the full KGI model infers that the entity `bell_centre`, a sports complex in Montreal more commonly referred to following the French convention, as ”Centre Bell”, is a building and attraction. While these inferences show the importance of entity resolution, the small difference in results suggest that the low coverage of entity resolution information in the dataset prevents a more significant impact on the results.

Finally, when rules for the ontological relationship are removed from the model, both the AUC and recall drop substantially, but precision increases. Analyzing the results of the two methods, we notice that the NoOnto method infers facts that violate ontological labes (as expected). An example of such an inference is that the entity

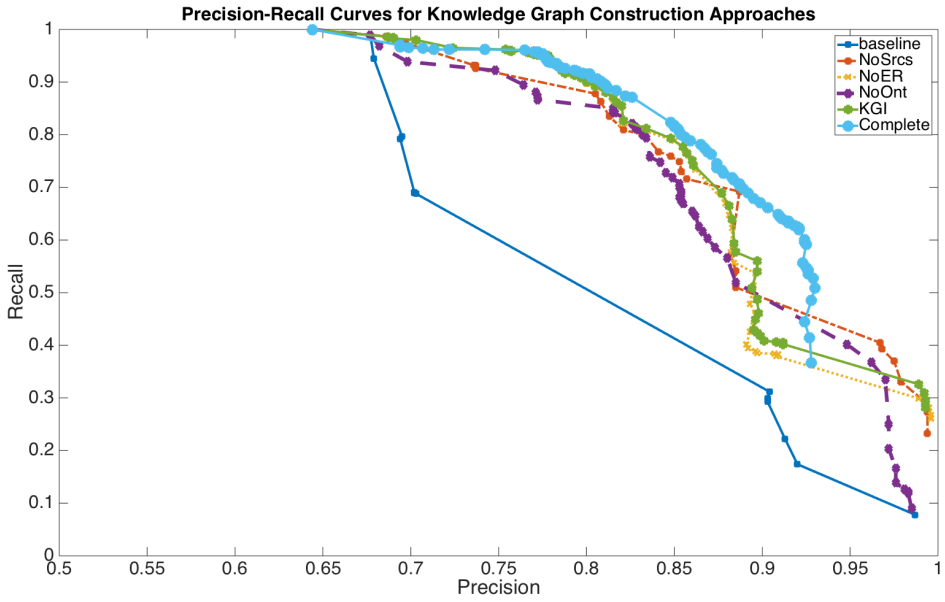


Figure 4.2: This figure shows the precision-recall curve for the different knowledge graph construction models. The baseline model, which does not use any collective reasoning, severely underperforms all other approaches. Models that omit the confidence information of uncertain sources (orange squares), entity resolution (yellow exs), or ontological information (purple diamonds) do not perform as well as knowledge graph identification (green hexagons). Complete knowledge graph identification (blue circles) in the open-world setting, infers a complete knowledge graph and shows superior performance, except at high precision cutoffs.

`community_college` plays the sport baseball. However, the overgeneralization of community colleges prevents this fact from being a useful addition to the knowledge base. In contrast, the full KGI model is able to learn from existing relations and infer labels for the entities. For example, the KGI model infers that `comiskey_park` and `cardiff_international_arena` are both locations, presumably using domain and range relationships from other facts.

Table 4.5: Producing a complete knowledge graph reduces performance on the test set, suggesting that the true complexity of the problem is masked when generating a limited set of inferences.

Method	AUC	Prec	Recall	F1
NELL	0.765	0.801	0.580	0.673
PSL-KGI-Complete	**0.891**	**0.825**	**0.872**	**0.848**
PSL-KGI	0.904	0.767	0.956	0.851

4.4.6 Results for Open-World Settings

One drawback of our comparisons to previous work is the restriction of the model to a small set of inference targets. The construction of this set obscures some of the challenges presented in real-world data, such as conflicting evidence, as well as strengths from collective modeling, such as supportive inferences resulting from parsimony in a large network of facts. For a fuller exploration of the knowledge graph construction problem, we assess the performance of our method in a setting without predefined inference targets. In this setting, the inference problem does not restrict potentially contradictory inferences or supportive inferences. We ran knowledge graph identification using the same learned weights as the closed-world setting, and allowed lazy inference to produce a complete knowledge graph, as detailed in subsection 4.4.3. Results for this setting are shown in Table 4.5, with more detailed results in Appendix B.

The resulting inference produces a total of 4.9M total facts, which subsumes the test set. We report results on the test set as PSL-KGI-Complete. Allowing the model to optimize on the full knowledge graph instead of just the test set reduced the performance as measured by the particular test set relative to the KGI model in the closed-world set-

ting, suggesting that the noise introduced by conflicting evidence does have an impact on results. However, when comparing to the only other method that operates in the open-world setting, the NELL fact promotion strategy, our model improves all performance metrics and has substantially higher recall. One reason the NELL promotion strategy may perform poorly is that it attempts to restrict newly inferred facts to those consistent with its existing knowledge base, and does not allow revising existing beliefs. As a result, erroneous facts introduced early in the knowledge base construction process can hamper future efforts to expand the knowledge base. In contrast, our approach puts new extractions in the same context with facts learned earlier, and if a preponderance of evidence supports the new extractions over the old facts, the inferred truth value can remove such erroneous facts.

We also compare two knowledge bases learned by the open-world version of KGI, KGI-Complete, and the closed world version, KGI. A number of facts that have high truth values in the KGI-Complete model are missing from the KGI model because they are out of the scope of the testing set. Prominent examples include REL(lemur, jamie_callan, mlsoftwareauthor) and REL(bruins, banknorth_garden, teamhomestadium). In contrast, a handful of facts that have high scores in the KGI model are not inferred or have low truth values in the KGI-Complete model. Some of these facts seem to be useful additions to the knowledge graph, for example REL(rfk_memorial_stadium, washington, stadiumincity) and REL(lusaka, zambia, cityincountry). However, a number of erroneous facts also appear in this list, such as REL(boston_celtics, los_angeles, teamplaysincity), REL(pittsburgh_steelers,

baltimore, teamplaysincity) as well as overgeneralizations such as REL(georgia_tech, basketball, teamplayssport). In summary, the differences between the two approaches are a mixed bag; KGI-Complete infers facts beyond the scope of the narrowly defined test set, and achieves the greater coverage desirable in knowledge graph construction, while also weeding out some clearly erroneous facts, but along the way also misses a number of valid facts.

Fig. 4.2 highlights one of the shortcomings of the complete knowledge graph inference setting – the model has difficulty producing facts when a high-precision cutoff is necessary. At the highest truth value threshold, KGI-Complete achieves a precision of 0.93 (and recall of 0.37), while the KGI model from the closed world setting has a precision of 0.99 (and a recall of 0.28). One reason for the lower precision may be that the KGI model is trained specifically to maximize the performance on a small network of facts, while no similarly straightforward training objective can be constructed for the open-world setting. A second issue is that many facts restricted from closed-world inference can appear with small truth values in open-world inference, lowering the truth values of inferred facts and preventing the same stratification of extremely high truth values that occur in the closed-world setting.

4.5 Discussion

In this chapter, we show how to address the requirements of the knowledge graph identification problem setting. Our contributions are (1) implementing straightforward and interpretable model that performs the essential tasks in graph identification – entity resolution,

collective classification, and link prediction; (2) incorporating the important properties of the input data used to construct a knowledge graph: uncertainty and statistical features from the information extraction system and logical constraints between facts found in the ontology; (3) demonstrating how the model we have developed can be transformed into a probability distribution over possible knowledge graphs; (4) tackling the practical challenges in inferring knowledge graphs such as weight learning and open-world problem settings; and (5) performing extensive evaluation of our knowledge graph identification model on multiple datasets that demonstrates the value of the various components of the model, compares to existing state-of-the-art methods, and validates the hypothesis of knowledge graph identification. We also identify a number of promising extensions to this work, such as modeling a richer set of ontological constraints mentioned in Section 3.1, different and possibly more sophisticated approaches to weight learning, and deeper models for entity resolution, which we tackle in the next chapter.

Chapter 5: Entity Resolution for Knowledge Graphs

In Chapter 3 we provided examples of entity ambiguity in knowledge extraction systems, which motivated the problem of entity resolution. Next, in Chapter 4 we demonstrated how ontological knowledge of co-referent entities or measures of entity similarity can be exploited to improve the results of knowledge graph construction. The problem of entity resolution has been a longstanding challenge that has lead to significant research in many communities, including databases, information retrieval, and natural language processing.However, entity resolution in knowledge graphs presents additional opportunities, complexities and challenges. We analyze two key facets of entity resolution problems arising from the structure of knowledge graphs: using knowledge graph features and supporting collective dependencies in co-reference judgments. In this section, we discuss the problems confronting entity resolution in knowledge graphs and develop a general model adaptable to many entity resolution tasks and scenarios.

5.1 Problem Definition

The general problem of entity resolution is to take a set of references, such as diverse string representations of names, and produce a mapping from these references to entities, which represent a single concept. Entity resolution can be formulated as a clustering

problem, where each cluster of references represents an entity, or as a pairwise matching problem, where two references are assessed for equality and a connected component of equal references represents an entity. In both formulations, a key problem is measuring the similarity of references, either to determine cluster coherence or to produce pairwise co-reference predictions.

The earliest entity resolution research focused on developing specialized similarity measures for strings and attributes. More recent work in entity resolution has focused on using relationships between references to generate *relational* features. For example, Bhattacharya and Getoor (2007) introduce relational features and similarities, and using a collective relational clustering approach, demonstrate superior results to non-collective approaches. One key requirement for knowledge graph entity resolution is the ability to translate knowledge graph features, such as attributes, types, and the many different relationships between entities, into features that can be used to determine the similarity of references.

A second key requirement for entity resolution in knowledge graphs is correctly handling collective dependencies in entity resolution decisions. Entity resolution problems are inherently collective due to transitivity or functionality constraints of equality. More formally, when resolving a set of references, a transitivity constraint requires that if A and B are co-referent, and B and C are co-referent, then A and C. A functionality constraint can exist in a setting where a bijective mapping between references in two sets, **S** and **T**, is desired, if A $\in$ **S** and B $\in$ **T** are co-referent, then, for all C $\in$ **T**, A and C cannot be co-referent. While transitivity and functionality are standard examples of collective entity resolution challenges, the knowledge graph setting often includes more

sophisticated examples of collective reasoning. For example, if we have two knowledge graphs that include references with relations pertaining to genealogical information, we might have references such as: REL(E_1, O_1, `parent`), REL(E_2, O_2, `parent`), then determining that E_1 and E_2 are co-referent can provide useful information that $O_1$and $O_2$are potentially co-referent as well.

This discussion hints at the diversity of entity resolution problems in knowledge graphs. Different phases of knowledge graph construction may face unique entity resolution challenges. We enumerate three general cases where entity resolution is necessary in knowledge graphs. Entity resolution may be required to:

1. resolve ambiguity in a set of candidate extractions
2. incorporate new extractions into an existing knowledge graph
3. combine information from two or more knowledge graphs

We discuss each of these scenarios in detail in the following paragraphs.

5.1.1 Ambiguity In Candidate Extractions

In previous chapters, the emphasis has been on creating a knowledge graph using a set of candidate facts generated by information extraction techniques. These information extraction techniques are subject to many sources of ambiguity. Each technique may process the same information differently, yielding many references from the same source material. Furthermore, the extraction source material may be inherently ambiguous, using different references for the same entity within a document, such as partial names or titles. Another common problem is anaphora, such as when a pronoun is used to with an ambiguous referent. Finally, the extractions are drawn from a corpus of documents, and

each document may have variations in the representation of references, such as alternate spellings, prefixes, suffixes, and abbreviations. As we have seen, in addition to the noise in entity references, noise also exists in attributes and relations ascribed to each reference. In this scenario, the goal is to cluster a set of noisy references with noisy attributes and relations into a coherent set of entities.

5.1.2 Incorporating New Extractions Into a Knowledge Graph

A somewhat simpler problem is extending an existing knowledge graph using new extractions. In this setting, the goal is to map each reference to an existing entity in the knowledge graph, or introduce a new entity into the knowledge graph. One strategy for dealing with new entities that do not exist in the knowledge graph is skolemization, where each potential new entity is given a unique identifier. References can now be matched with existing entities or the new, skolemized entities in the knowledge graph, casting the problem into the well-studied task of surjective bipartite matching from references to entities.

Through this formulation, the added constraint that each reference must match a single entity can often simplify the entity resolution process. While the attributes and relationships of the extracted reference may be noisy, as motivated in the previous scenario, the attributes and relationships of entities in the knowledge graph are expected to be highly reliable. As a result, relational features and attribute similarity play a more significant role in determining whether a reference can be resolved to an existing entity in the knowledge graph, or due to conflicting information, the reference should be added as a new entity with different attributes and relations.

5.1.3 Combining Information From Multiple Knowledge Graphs

The final knowledge graph entity resolution scenario adheres most closely to the traditional approaches to entity resolution, where the goal is to combine information from two or more databases. In this setting, the goal is to find a mapping between entities in knowledge graphs, and then combine the attributes and relations of these entities. This problem can be formulated as mapping each knowledge graph to a "canonical" knowledge graph or instead be cast as a pairwise matching task between each pair of knowledge graphs. The latter formulation can introduce additional complexity in the form of transitivity constraints for equality across all knowledge graphs. These constraints can add new features for entity resolution, but may also make the desired mapping more computationally demanding. A further complication in this setting is that the knowledge graphs may use different schemas and ontologies. This problem is not covered in this chapter, but the development of standard ontologies and the problems of ontology matching or schema mapping have been extensively researched.

While these three entity resolution settings each present unique challenges, our goal is to provide a unified model for entity resolution. The goal of this model is to adapt to the diverse circumstances present in knowledge graph construction tasks. In the next section, we outline the structural elements of this model, and then introduce a probabilistic model for entity resolution that incorporates these elements into an entity resolution system.

5.2 Approach

The crucial aspect that distinguishes knowledge graphs from standard entity resolution problems is the rich and regular structure of the knowledge graph. Our goal is to leverage this structure to build an entity resolution model that is easy to understand and customize, while still capturing the rich information present in the knowledge graph. We consider two dimensions to the entity resolution model: feature granularity and collective inference. First, we organize the features in knowledge graphs based on the granularity of knowledge required. While the most basic features rely on string similarity or generic rules of functionality and transitivity, more complicated features involve new entities, attribute similarity, equivalence classes of relations, and domain-specific patterns. Each of these features can be classified as local (involving a single co-reference decision) or collective (imposing a dependency between two or more co-reference decisions). Table 5.1 summarizes the knowledge graph features used by the entity resolution methods, and the following subsections delve more deeply into each of these feature sets. For each type of feature, we provide examples of corresponding logical rules. These rules can be combined in a probabilistic modeling framework, such as PSL, to produce a collective probabilistic graphical model for entity resolution.

5.2.1 Local and Collective Knowledge Graph Features

As motivated earlier, there are two broad classes of features in knowledge graphs: local and collective. Local features are those that can be computed for a pair of entities (or references) independently of the co-reference decisions of other entities in the knowledge

Table 5.1: Knowledge graph features categorized based on collective dependencies and level of granularity

	local	collective
basic	similarity scores	transitive, functional, sparsity
new entity	new entity prior	new entity penalty (sparsity)
abstract KG	type matching, type penalty	relation matching/equivalence
domain-specific	restricted type matching	restricted relation matching

graph. Examples of local features include string similarity of names, image similarity of photographs, type agreement, and attribute agreement. One key characteristic for a local feature is that its value does not depend on the entity resolution decisions for other pairs of entities. This characteristic allows local features to be computed once for a pair of features and reused. Consequentially, relying on local features for entity resolution can decrease computational overhead and improve entity resolution performance.

In contrast to local features, collective features involve dependencies between co-reference decisions, and due to these dependencies are more difficult to compute. The transitivity and functionality constraints in Section 5.1 are examples of common collective features that have been used in entity resolution. However, the structure of knowledge graphs allow many more collective features to be generated using relationships between entities. Knowledge graph features can be abstract, such as the overlap of object-arguments for a reference's relations, or very concrete, such as the link between parents and children in the earlier example.

5.2.2 Knowledge Graph Models at Different Granularity

In this section, we develop components for a knowledge graph entity resolution model. The components have been classified into four categories:

1. **basic** features that are common to all entity resolution scenarios
2. **new entity** features that helpful when adding new entities into a knowledge graph
3. **abstract KG** features that are universal across many knowledge graph structures
4. **domain-specific** features that are designed to resolve a particular class of entities

In the subsequent sections, we will introduce logical rules for each type of feature, distinguishing between local and collective rules. The goal of these rules is to determine a pairwise resolution between two entities, denoted by SAME(E_1, E_2) for entities $E_1$and E_2. Note that the SAME predicate is distinct from the SAMEAS predicate, which is used to capture ontological information, such as `owl:sameAs`.

Since knowledge graphs routinely contain millions of entities, assessing pairwise equality between all entities is infeasible. A common technique to avoid the polynomial explosion of entity matching is **blocking**, which uses a simple heuristic to produce potential entity matches. Using this smaller set of possible resolutions can substantially improve scalability. In the following rules, we will represent a blocked pair of entities with the predicate CANDSAME. Blocking can also be used to restrict matches based on the entity resolution scenario. For example, in Section 5.1.2 where the goal is to map references in a set of extractions to an existing knowledge graph, blocking can be used to scope entity resolution to only allow matches between extractions and the knowledge graph, disallowing matches within the extractions or within the knowledge graph.

5.3 Modeling Knowledge Graph Entity Resolution

5.3.1 Basic Features

Rules for Local Features

Basic features are those common to all entity resolution scenarios, such as similarity functions and prior probabilities. we introduce three rules corresponding to basic local features. Rule 5.1 and Rule 5.2 are priors for SAME. Often, a negative prior (5.1) is useful to implement a default policy that entities are not co-referent unless supported by evidence. A positive prior can also be useful in some models to establish a baseline match confidence for two entities that have been blocked.

The final basic local rule uses a similarity function, SIM, to assess whether two entities are co-referent. In general, the similarity function can depend on the representation of the entities (e.g. images, sound files, or textual representations). A great deal of research in entity resolution has been devoted to designing effective similarity functions for entity resolution. Examples of popular similarity functions are Levenstein (Navarro, 2001; Wagner and Fischer, 1974), Jaro (Jaro, 1995), Jaro-Winkler (Winkler, 1999), Monge-Elkan (Elkan and Monge, 1996), Fellegi-Sunter (Fellegi and Sunter, 1969), Needleman-Wunsch, and Smith-Waterman (Durbin et al., 1998). In Rule 5.3 the similarity function is not explicitly specified, but a popular similarity function or combination of functions (Bilenko and Mooney, 2003) can be used to determine similarity.

$$\neg \text{SAME}(E_1, E_2). \qquad (5.1)$$

$$\textsc{CandSame}(E_1, E_2) \Rightarrow \textsc{Same}(E_1, E_2). \tag{5.2}$$

$$\textsc{CandSame}(E_1, E_2) \wedge \textsc{Sim}(E_1, E_2) \Rightarrow \textsc{Same}(E_1, E_2). \tag{5.3}$$

Rules for Collective Features

The collective basic features incorporate the fundamental properties of equality: symmetry (Rule 5.4) and transitivity (Rule 5.5). Symmetry enforces the constraint that the order of the arguments to SAME do not matter. Transitivity, discussed in Section 5.1, ensures that the co-reference process generates tight clusters of entities by encouraging co-reference cliques. Finally, Rule 5.6 has an opposite effect, encouraging sparsity by promoting functionality for the SAME predicate. Not all entity resolution scenarios require functionality for co-references, but those discussed in Section 5.1.2 and Section 5.1.3 can benefit from such constraints.

$$\textsc{Same}(E_1, E_2) \Rightarrow \textsc{Same}(E_2, E_1). \tag{5.4}$$

$$\textsc{CandSame}(A, B) \wedge \textsc{CandSame}(B, C) \wedge \textsc{CandSame}(A, C) \wedge \textsc{Same}(A, B) \wedge \textsc{Same}(B, C) \Rightarrow \textsc{Same}(A, C). \tag{5.5}$$

$$\textsc{CandSame}(A, B) \wedge \textsc{CandSame}(A, C) \wedge \textsc{Same}(A, B) \Rightarrow \neg\textsc{Same}(A, C). \tag{5.6}$$

5.3.2 New Entity Features

Rules for Local Features

In problem settings where entity resolution is matching with respect to an existing knowledge graph, such as Section 5.1.2 and Section 5.1.3, the appropriate entity may not exist in the target knowledge graph. In these settings, we generate a new entity placeholder for each source reference. This placeholder will have no inherent relations, types, or attributes and will have a default similarity value. we designate these entities using the NEWENTITY predicate. Rule 5.7 establishes a prior that any reference matches a new entity. In subsequent rules, the NEWENTITY will be used to scope the rule to existing entities, which avoids penalizing new entity matches based on relations, types and attributes which are missing.

$$\text{CANDSAME}(E_1, E_2) \ \wedge \ \text{NEWENTITY}(E_1) \Rightarrow \ \text{SAME}(E_1, E_2). \tag{5.7}$$

Rules for Collective Features

While a prior can be helpful, the desired behavior in entity resolution systems is to add a new entity only when no other entity in the knowledge graph appears to match. Rule 5.8 prevents a new entity from matching when a previously existing entity is a strong match for a reference.

$$\begin{aligned}\text{SAME}(E_1, E_2) \ \wedge \ \text{CANDSAME}(E_1, E_3) \\ \wedge \ \text{NEWENTITY}(E_3)\end{aligned} \Rightarrow \ \neg\text{SAME}(E_1, E_3). \tag{5.8}$$

5.3.3 Abstract Knowledge Graph Features

Abstract knowledge graph features use the relational structure and attributes shared by all knowledge graphs. The key strength is that these features are broadly applicable to any knowledge graph entity resolution problem. In scenarios such as Section 5.1.1, abstract knowledge graph rules can be used to collectively infer relations and labels in the knowledge graph while simultaneously determining entity co-references. However, one drawback of abstract knowledge graph rules is that their broad applicability may limit their usefulness. Rules that are agnostic to the particular labels and relations in a knowledge graph may have difficulty prioritizing which labels and relations are useful for entity resolution. One potential solution to this issue when ample training data is available is to introduce rules and then learn rule weights for each label and relation separately.

Rules for Local Features

Knowledge graph entities have associated properties such as attributes, labels, and type information that provide the basis for local features. Rule 5.9 specifies that these properties agree for two entities. Since many potential candidate matches may share the same properties, the rule is mediated by the candidate similarity, supporting similar matches more strongly than dissimilar matches. While entities with agreeing properties are a signal of co-reference, properties that are missing or explicitly disagree can be strong signals against co-reference. Rule 5.10 requires that co-referent entities share properties, but provides an exception for new entities which lack properties. Note that a symmetric rule for the second entity is not shown. These rules are most useful in entity resolution settings where knowledge graph information is relatively complete, whereas noisy or incomplete

extractions may hamper entity resolution. Rule 5.11 provides a stronger signal by incorporating the knowledge graph ontology, disallowing entities with mutually-exclusive properties from matching. Even when extractions are noisy and properties incomplete, this signal can provide strong evidence against a potential co-reference match.

$$\begin{aligned} \textsc{CandSame}(E_1, E_2) \;&\wedge\; \textsc{Sim}(E_1, E_2) \\ &\wedge\; \textsc{Lbl}(E_1, L) \\ &\wedge\; \textsc{Lbl}(E_2, L) \\ &\Rightarrow\; \textsc{Same}(E_1, E_2). \end{aligned} \tag{5.9}$$

$$\begin{aligned} \textsc{CandSame}(E_1, E_2) \;&\wedge\; \textsc{Lbl}(E_1, L) \\ &\wedge\; \neg\textsc{Lbl}(E_2, L) \\ &\wedge\; \neg\textsc{NewEntity}(E_2) \\ &\Rightarrow\; \neg\textsc{Same}(E_1, E_2). \end{aligned} \tag{5.10}$$

$$\begin{aligned} \textsc{CandSame}(E_1, E_2) \;&\wedge\; \textsc{Lbl}(E_1, L_1) \\ &\wedge\; \textsc{Lbl}(E_2, L_2) \\ &\wedge\; \textsc{Mut}(L_1, L_2) \\ &\Rightarrow\; \neg\textsc{Same}(E_1, E_2). \end{aligned} \tag{5.11}$$

Rules for Collective Features

The collective abstract knowledge graph entity resolution parallel the local rules, but operate over relations and involve pairs of co-referent entities. Rule 5.12 requires that two co-referent entities have the same relation with co-referent objects. The collective nature of the rule introduces a dependence between entities that participate in the same relation across knowledge graphs, supporting co-references between the subjects and objects of the relation. Rule 5.13 has the opposite effect, penalizing co-references for matches between existing entities that do not share the same relations. Echoing the previous remarks on knowledge graph rules, these rules have limited usefulness in noisy or incomplete

knowledge graphs where many relations may be missing. However, Rule 5.14 uses the ontology to find a stronger signal in mutually-exclusive relations.

$$\begin{aligned} \textsc{CandSame}(E_1, E_2) \;&\wedge\; \textsc{CandSame}(O_1, O_2) \\ &\wedge\; \textsc{Sim}(E_1, E_2) \\ &\wedge\; \textsc{Same}(O_1, O_2) \\ &\wedge\; \textsc{Rel}(E_1, O_1, R) \\ &\wedge\; \textsc{Rel}(E_2, O_2, R) \\ &\Rightarrow\; \textsc{Same}(E_1, E_2). \end{aligned} \tag{5.12}$$

$$\begin{aligned} \textsc{CandSame}(E_1, E_2) \;&\wedge\; \textsc{CandSame}(O_1, O_2) \\ &\wedge\; \textsc{Same}(O_1, O_2) \\ &\wedge\; \neg\textsc{Rel}(E_1, O_1, R) \\ &\wedge\; \neg\textsc{NewEntity}(E_1) \\ &\wedge\; \neg\textsc{NewEntity}(O_1) \\ &\wedge\; \textsc{Rel}(E_2, O_2, R) \\ &\Rightarrow\; \neg\textsc{Same}(E_1, E_2). \end{aligned} \tag{5.13}$$

$$\begin{aligned} \textsc{CandSame}(E_1, E_2) \;&\wedge\; \textsc{CandSame}(O_1, O_2) \\ &\wedge\; \textsc{Same}(O_1, O_2) \\ &\wedge\; \textsc{Rel}(E_1, O_1, R) \\ &\wedge\; \textsc{Rel}(E_2, O_2, S) \\ &\wedge\; \textsc{RMut}(R, S) \\ &\Rightarrow\; \neg\textsc{Same}(E_1, E_2). \end{aligned} \tag{5.14}$$

5.3.4 Domain-Specific Knowledge Graph Features

While abstract knowledge graph features provide a generally-applicable tool for knowledge graph entity resolution, in many cases domain experts can rely on experience to assess the most important features for matching knowledge graphs. Since our model uses interpretable rules that are easy to generate, domain experts can easily add and remove

rules to the model to capture the most relevant relationships. In this section, we provide some example rules for the task of matching knowledge graphs in the music domain. These rules are derived from rules used in an industry knowledge graph matching system, supporting the assertion that rules are a natural and common form of supplying domain expertise for knowledge graphs.

Rules for Local Features

One example of a domain rule that strongly supports co-reference are relations with categorical domains. The `release_type` relation in musical domains differentiates between singles, EPs, and albums. Since the domain of the relation is a small, enumerated set, matching release types across co-references is important. Rule 5.15 incorporates this domain knowledge in a rule for release type matching. Just as some relations are more important than others, so are types, attributes and labels. While general purpose ontologies have a `person` type, a more specific type can be far more useful in matching. Rule 5.16 provides a special case for `artist`, a subtype of `person`. One way of interpreting this rule is a type-specific prior for entity matches. By choosing appropriate weights, these rules can also moderate the importance of a similarity metric in a particular domain. For example, a high similarity value may not be meaningful for a broad domain (e.g. `person`) but can provide a stronger disambiguating signal for a more selective domain (e.g. `artist`).

$$\begin{aligned} \textsc{CandSame}(E_1, E_2) \;&\wedge\; \textsc{Sim}(E_1, E_2) \\ &\wedge\; \textsc{Rel}(E_1, L, \texttt{release\_type}) \\ &\wedge\; \textsc{Rel}(E_2, L, \texttt{release\_type}) \\ &\Rightarrow\; \textsc{Same}(E_1, E_2). \end{aligned} \tag{5.15}$$

$$\begin{aligned} \textsc{CandSame}(E_1, E_2) \;&\wedge\; \textsc{Sim}(E_1, E_2) \\ &\wedge\; \textsc{Lbl}(E_1, \texttt{artist}) \\ &\wedge\; \textsc{Lbl}(E_2, \texttt{artist}) \\ &\Rightarrow\; \textsc{Same}(E_1, E_2). \end{aligned} \tag{5.16}$$

Rules for Collective Features

Similarly, domain experts can select the most important relations for resolution in a domain. Rule 5.17 which focuses on co-referent releases of the same album can be more useful than a rule which focuses on `release_label` since a label typically has many releases. Domain rules can also incorporate more complex criteria. Rule 5.18 has a similar form to normal collective relational rules, but includes an additional constraint that the albums and artists must all come from the same genre.

$$\begin{aligned} \textsc{CandSame}(E_1, E_2) \;&\wedge\; \textsc{Sim}(E_1, E_2) \\ &\wedge\; \textsc{CandSame}(O_1, O_2) \\ &\wedge\; \textsc{Rel}(E_1, O_1, \texttt{release\_album}) \\ &\wedge\; \textsc{Rel}(E_2, O_2, \texttt{release\_album}) \\ &\wedge\; \textsc{Same}(E_2, E_1)) \\ &\Rightarrow\; \textsc{Same}(O_1, O_2). \end{aligned} \tag{5.17}$$

$$\begin{aligned} \textsc{CandSame}(E_1, E_2) \;&\wedge\; \textsc{CandSame}(O_1, O_2) \\ &\wedge\; \textsc{Sim}(E_1, E_2) \\ &\wedge\; \textsc{Sim}(O_1, O_2) \\ &\wedge\; \textsc{Rel}(E_1, O_1, \texttt{album\_artist}) \\ &\wedge\; \textsc{Rel}(E_2, O_2, \texttt{album\_artist}) \\ &\wedge\; \textsc{Rel}(E_1, G, \texttt{album\_genre}) \\ &\wedge\; \textsc{Rel}(E_2, G, \texttt{album\_genre}) \\ &\wedge\; \textsc{Rel}(O_1, G, \texttt{artist\_genre}) \\ &\wedge\; \textsc{Rel}(O_2, G, \texttt{artist\_genre}) \\ &\wedge\; \textsc{Same}(O_1, O_2) \\ &\Rightarrow\; \textsc{Same}(E_1, E_2). \end{aligned} \tag{5.18}$$

Table 5.2: Summarizing entity resolution rules and matching them to application

	Local/ collective (5.2.1)	New extractions (5.1.1)	Extend KG (5.1.2)	Multiple KGs (5.1.3)
Negative prior (5.1)	L	Y	Y	Y
Positive prior (5.2)	L	Y	Y	Y
Similarities (5.3)	L	Y	Y	Y
Symmetry (5.4)	C	Y	Y	Y
Transitivity (5.5)	C	Y	N	Y
Sparsity (5.6)	C	N	N	Y
New Entity prior (5.7)	L	N	Y	Y
New Entity penalty (5.8)	C	N	Y	Y
Label agreement (5.9)	L	N?	Y	Y
Label disagreement (5.10)	L	N?	Y?	Y?
Label exclusion (5.11)	L	Y	Y	Y
Relational agreement (5.12)	C	N?	Y	Y
Relational disagreement (5.13)	C	N?	Y?	Y?
Relational exclusion (5.14)	C	Y	Y	Y
Domain-specific categorical relations (5.15)	L	Y	Y	Y
Domain-specific prior (5.16)	L	Y	Y	Y
Domain-specific relations (5.17)	C	Y?	Y	Y
Domain-specific relational criteria (5.18)	C	Y?	Y	Y

5.3.5 Synthesis

The previous section introduced a number of rules for entity resolution, categorized by whether the rule used local or collective information and the granularity of the knowledge graph features used. In the discussion of each rule, we referenced the three knowledge graph entity resolution scenarios (introduced in sections 5.1.1, 5.1.2, and 5.1.3) and the conditions under which the rule was applicable to the scenario. The rules and this discussion is summarized in Table 5.2. Note that some of the entries have question marks, which reinforce the guidance that the corresponding rules may be appropriate based on dataset characteristics such as noise and sparsity.

The knowledge graph entity resolution model presented in this section is a general and versatile approach to entity resolution in richly structured domains. Since the requirements of different entity resolution scenarios vary, care must be taken to select the appropriate rules and design meaningful domain-specific rules. However the proliferation of domain-specific entity resolution methods (Durbin et al., 1998; Winkler, 1999) and anecdotal evidence from many projects in industry suggest that many bespoke entity resolution systems are already in use. Despite the widespread use of such systems and substantial research in entity resolution, no general-purpose, collective framework has been adopted across domains.

This chapter provides a general guide to designing entity resolution systems for knowledge graphs. The rules presented can be used as templates for many approaches that jointly model entity resolution decisions, such as linear programs and probabilistic models. We use the rules as the basis for a probabilistic soft logic (PSL) program for

performing entity resolution. PSL is a natural choice for entity resolution models, since entity resolution models have many collective dependencies, use real-valued similarity measures, and often include a vast number of entities.

5.4 Evaluation

We evaluate our knowledge graph entity resolution approach on two very different datasets from different entity resolution scenarios. The first dataset, corresponding to the scenario in Section 5.1.1, involves clustering unresolved references with associated relations and attributes from different web sources. The second dataset, corresponding to the scenario in Section 5.1.3, involves resolving entities between the MusicBrainz music knowledge graph and the Freebase broad-coverage knowledge graph.

NELL

NELL extracts a series of facts from text, and uses a set of heuristics to map textual references to entities. This entity mapping process includes two phases: first, textual references are clustered to identify senses and then these textual references are mapped to the appropriate senses. The entity mapping process does not use the context of the knowledge graph, which can improve the performance on entity mapping. Furthermore, the entity mapping process does not attempt to perform entity resolution between different textual references that refer to the same underlying entity.

In order to investigate the effectiveness of entity resolution applied to ambiguous candidate extractions, We worked with the NELL team to collect data from a new NELL

instance that performed only the first phase of entity mapping, clustering textual references to generate senses. The second phase of entity mapping was not performed, so this NELL instance produced raw candidate extractions in terms of the original textual references. The goal in this setting is to collectively determine the facts in the knowledge graph along with the entity co-references.

NELL's Entity Resolver produces match scores for pairs of textual references. We extend these match scores by computing a number of string similarity metrics for each pair of textual references, using the SecondString library (Cohen et al., 2003). The set of string similarities includes the Jaccard overlap (of characters), Jaro, Jaro-Winkler, Levenshtein, Monge-Elkan, and Smith-Wasserman similarity functions. These string similarities constituted local features for entity resolution.

Using data from the first iteration of NELL yielded 145K candidate relations, 200K candidate labels, 170K unique textual references that mapped to 190K potential entities. The NELL EntityResolver candidate generation produced 4K potential entity co-references. Since the dataset was collected from a new NELL instance, no existing entity match information was used or available. Furthermore, since there were no pre-existing entities, each textual reference was considered unknown and no special handling of new entities was required.

NELL does not have a reliable source of entity resolution data, so we manually labeled entity co-references. For each method, we chose the top 50 as well as 50 randomly selected entity co-references from each method. This selection process yielded 421 co-references after duplicates were removed. We then removed the truth values and randomized the order of the co-references for judging.

Method	AUPRC	F1	Prec.	Recall
Basic, Local	0.267	0.333	0.214	0.759
Basic & KG, Local	0.247	0.426	0.298	0.747
Basic, All	0.307	0.446	0.333	0.675
Basic & KG, All	**0.351**	**0.453**	0.383	0.554

Table 5.3: Comparing the performance of knowledge graph entity resolution rules in for the NELL dataset. Performance improves by adding knowledge graph features and collective features, with the best performance with both.

Entities were judged to be co-referent when there was an unambiguous interpretation of the textual references that corresponded to one entity. This, for example, excludes "Giants" matching "San Jose Giants" since many other sportsteams share the same name. Similarly, when a textual reference was the amalgamation of two entities, matches with either entity were dissolved. For example, this invalidates "Quito" from matching with "Quito and Cuenca". However, merged entities were judged co-referent, allowing "Konica" and "Konica Minolta" to be co-referent since the company Konica merged with Minolta to become the merged company.

Results for the NELL entity resolution are reported in Table 5.3. The baseline, Basic-Local entity resolution uses only priors and the various similarity metrics.

MusicBrainz and the Google Knowledge Graph

The second dataset for entity resolution involved mapping entities between two knowledge graphs. The first knowledge graph was from the MusicBrainz music knowledge base, introduced in the previous chapter. The second knowledge graph was the Google Knowledge Graph. We restricted the entities from the Google Knowledge Graph in our dataset to select only those entities that were in the publicly available Freebase knowledge

base.

An existing, proprietary pipeline to map entities between these two knowledge graphs exists. The pipeline uses Boolean rules restricted to discrete features. The system is designed to consider entity resolutions sequentially, and as a result cannot use all collective information between resolution decisions. When a match decision for an entity cannot be made by the pipeline, the entity is manually resolved by a human annotator. Evaluation of the existing pipeline showed a high error rate, while manually annotated entities contained no errors. Our experiments focus on the entities that were not successfully matched using the existing pipeline, which constitute the most difficult entity resolution decisions.

We begin with a dataset of 11K entities added to the MusicBrainz knowledge graph between 5/5/2014 and 6/29/14 that were manually annotated and have reliable ground truth. We identify 332K Freebase entities that are potential candidate matches for the MusicBrainz entities using a string similarity measure that is normalized for entity frequency. Since these newly added entities are often not found in Freebase, we generate new entity candidates for each MusicBrainz entity. The entity resolution dataset also includes 1M known entity mappings between the two knowledge graphs and 15.7M relations between entities.

Table 5.4 summarizes the results of these experiments. The baseline method uses only local rules, and achieves an area under the precision-recall curve (AUPRC) of 0.416 and an F1 measure of 0.734. Adding collective rules and domain-specific features that use the knowledge graph improves performance, with an AUPRC of 0.569 and an F1 of 0.805. Incorporating rules to handle new entities further improves performance with an

Method	AUPRC	F1	F1 (Exist)	F1 (New)
Basic & NewEntity, Local	0.416	0.734	0.169	0.744
Basic & Domain, All; NewEntity, Local	0.569	0.805	**0.305**	0.831
Basic & Domain & NewEntity, All	**0.724**	**0.840**	0.070	**0.895**

Table 5.4: Comparing the performance of knowledge graph entity resolution rules when merging MusicBrainz entities into the Knowledge Graph. Due to a skew toward new entities, the collective new entity rules dramatically improve overall performance, but with a substantial drop in the F1 measure for existing entities

AUPRC of 0.724 and an F1 measure of 0.840.

To better understand the performance, we separate the F1 measure for existing entities and new entities. In the dataset we collected, the entity mappings are skewed toward new entities, so that approximately 75% of entities in the MusicBrainz knowledge graph are not found in the Freebase entities. Thus the New Entity rules can have a dramatic influence on the performance by improving the performance on new entities while having a marked decrease in performance in existing entities.

5.5 Discussion

The growing importance of knowledge graphs has resulted in an increased emphasis on entity resolution for such structured domains. The collective relationships in a knowledge graph provide the key to improving the performance of entity resolution. In this chapter, we provided an inventory of important features necessary for entity resolution in the context of knowledge graphs and identified the corresponding knowledge graph settings

where these features are important. Building entity resolution models, particularly collective models, has required a great deal of time and effort. The general nature of this model makes it applicable to many different problem settings, and a PSL implementation of our entity resolution model makes it accessible for rapid prototyping and experimentation for a variety of entity resolution problems.

Chapter 6: Scaling Knowledge Graph Identification

One of the longstanding scalability challenges in artificial intelligence is tackling the combinatorial explosion, requiring optimization over an exponentially large space. Knowledge graph identification falls squarely into this space: a Boolean assignment of truth values to facts that satisfies a collection of constraints posed as logical formulas is an instance of the NP-complete maximum satisfiability (MAX-SAT) problem (Tsang, 1995). In this chapter, we analyze the theoretical complexity and empirical scaling performance of knowledge graph identification, and then develop a method to partition and distribute knowledge graph identification across machines.

6.1 Scalability Analysis of Knowledge Graph Identification

Two major forces contribute to the scalability challenges of knowledge graph identification: the state space of the optimization variables and the number of optimization terms. The state space of knowledge graph identification is the set of joint assignments to all facts in the knowledge graph. For a discrete model of a knowledge graph with F facts, the state space is 2^F, corresponding to a true or false assignment for each fact. In a continuous model, the state space is $O(\mathbb{R}^F)$, corresponding to an assignment in the $[0, 1]$ interval for each fact.

The second factor affecting scalability in knowledge graph construction is the number of optimization terms. Collective knowledge graph models incorporate constraints or rules relating the facts in the knowledge graph, and each of these ground constraints or rules is converted into an optimization term. The number of groundings of a rule is proportional to the number of atoms in the rule. Given a rule with p atoms, the number of ground rules is $O(F^p)$ yielding polynomial growth as the number of facts increase.

Even using simple, pairwise constraints ($p = 2$) such as domain, range, mutual exclusion, and subsumption can lead to a large blowup in the number of constraints. For more sophisticated knowledge graph models with rules relating many facts, the value of p can be much higher and produce a greater scalability challenge. While the worst case scaling is polynomial, in practice the growth in the number of groundings depends on the distribution of facts. A key consideration in determining how many optimization terms are required is the sparsity of the rules and constraints. For example, mutual exclusion constraints create a dense subgraph between all of the labels for a given entity.

These scalability considerations of the knowledge graph identification problem pose difficulties to many approaches inferring a knowledge graph. Search-based approaches have to contend with a vast state space. Developing usable search heuristics to explore this state space using the collection of rules and constraints in a knowledge graph may be difficult, due to the large number of ground terms. Optimization techniques that rely on sampling, such as Markov chain Monte Carlo, can be hampered by the large number of terms as well, since updates to the variable assignments depend on the ground rules and require recomputing the objective repeatedly. Additionally, the large state space can result in many local optima and make convergence difficult.

6.2 Scaling Knowledge Graph Identification with HL-MRFs

Our implementation of knowledge graph identification circumvents these issues through its choice of model and optimization. PSL defines a class of models known as hinge-loss Markov random fields (HL-MRFs). A key strength of HL-MRFs is that the inference objective is convex, allowing exact optimization, albeit in the approximate, soft-truth domain. The convexity of inference allows KGI to employ the many different approaches and algorithms to convex optimization developed over the last decades(Boyd and Vandenberghe, 2004).

The experiments in Section 4.4 show how the PSL implementation of KGI can handle problems from real-world datasets like NELL, which include millions of candidate facts. Inference when an explicit query set of 70K facts is given (PSL-KGI) requires a mere 10 seconds. The MLN method we compare against takes a few minutes to an hour to run for the same setting. When inferring a complete knowledge graph without known query targets, as in the LinkedBrainz and complete NELL experiments, inference with MLNs is infeasible. In contrast, knowledge graph identification on the NELL dataset can produce the complete knowledge graph containing 4.9M facts in only 130 minutes. The ability to produce complete knowledge graphs in these realistic settings is an important feature of our implementation of knowledge graph identification.

To better understand the scalability of KGI, we performed an extensive set of experiments by partitioning the NELL extractions while preserving groundings. We generated over 1000 knowledge graph identification problems and recorded the inference time for each of these problems (which includes costs related to grounding out the model). Fig.

6.1 shows this inference time scales with the size of the KGI problem, which is expressed in terms of the number of ground rules and constraints. Every data point on the scatterplot represents a single run of KGI inference. Although there are outliers, the vast majority of these KGI instances appear to scale linearly with the number of optimization terms.

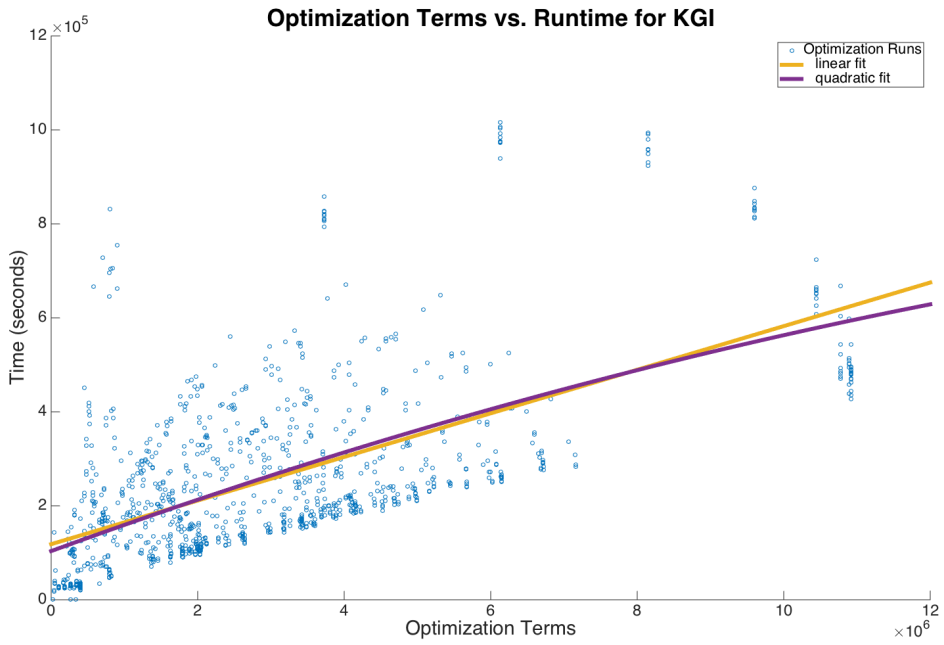


Figure 6.1: A plot capturing the running time and problem size (in terms of optimzation terms) of over 1000 KGI executions. Each execution is represented as a point in the scatter plot. Trendlines for a linear fit and quadratic fit of the relationship between problem size and running time are shown. Scalability of KGI appears to grow linearly with optimization terms.

6.3 Scalability Challenges for Knowledge Graph Identification

Our current approach for knowledge graph identification easily scales to millions of extractions. Unfortunately, even though knowledge graph identification models implemented as hinge-loss Markov random fields show impressive scaling performance as the

number of optimization terms grows, the analysis in Section 6.1 suggests that the number of terms may grow polynomially with the number of extractions. The combination of more powerful extraction techniques and the vast information available on the Internet means that knowledge graph construction systems are encountering an ever-increasing amount of data. These trends suggest that the true scale of KGI is billions of extractions or more, and even quadratic scaling of knowledge graph identification can prove to be too restrictive for practical applications. We address this challenge by developing a parallel approach to knowledge graph identification which distributes the probabilistic model across many different machines.

6.3.1 Partitioning Knowledge Graphs for Distributed Processing

The key to distributing knowledge graph identification is partitioning the knowledge graph across multiple machines. Horizontal partitioning, or splitting data into multiple sets which are processed independently, is non-trivial for joint inference problems such as KGI. Many appealing strategies for partitioning extractions involve partitioning the extractions directly. A simple approach could simply partition the extractions randomly. Unfortunately, such a simple approach will lose relationships between facts, creating subproblems that may turn out to be equivalent to the independent models presented in Chapter 4.

More sophisticated approaches could operate on the entities in the knowledge graph, for example by generating the extraction graph from a set of noisy extractions. Using this extraction graph, one could cluster the graph into separate components using a graph

clustering technique. One popular technique for clustering a graph into components is the minimum graph cut (Ford and Fulkerson, 1956). The problem of generating a minimum cut corresponds to removing a small number of edges from the original graph in order to generate one or more disconnected components of the graph. Since these components do not share any vertices, each can be processed independently.

Clustering the extraction graph offers some benefits. First, the approach is relatively straightforward to implement since many methods exist for graph clustering and finding minimum cuts in graphs (Karger and Stein, 1996). Second, the clustering captures the relational information that relates entities, which would result in a partitioning grouping semantically similar clusters of entities. However, a number of issues make such approaches intractable. First, such a partitioning requires partitioning a graph with many edges (possibly billions or trillions), which presents a substantial scalability challenge. Second, partitioning extractions directly does not preserve the ontological relationships that form a key ingredient for generating a consistent knowledge base. We present an alternative method, based on partitioning the concept graph, that addresses both of these drawbacks.

6.3.2 Scalability via Ontological Partitioning

We confront the problem of partitioning knowledge graph identification by identifying a key insight: since the concept graph contains the ontological relationships relating extractions in our model, and these relationships are the basis for knowledge graph identification, we can partition the concept graph and, by so doing, partition the probabilistic model

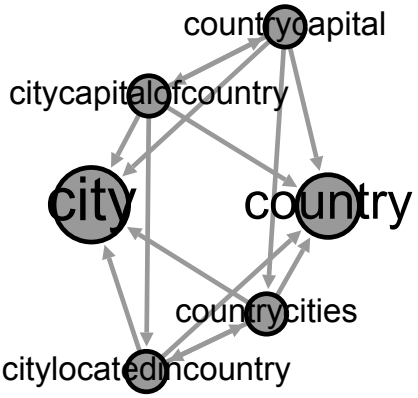


Figure 6.2: A small section of the NELL concept graph that shows the relationship between the labels city and country with relations such as capital.

generated by KGI. The concept graph, which was introduced in Section 3.1, has labels and relations from the knowledge graph as vertices and the edges connecting these vertices are the ontological constraints between these labels and relations. These ontological constraints were incorporated into the knowledge graph identification model presented in Section 4.2.

Our approach partitions the concept graph, which entails creating a partition of the labels and relations in the ontology. For example, the ontological relation DOM(`cityCapitalOfCountry`, `country`) would be converted to an edge of type DOM between the relation vertex `cityCapitalOfCountry` and the label vertex `country`. We show a small subset of the ontological graph for NELL in Figure 6.2. A partitioning of the concept graph might separate the `city` and `country` labels and related relations into distinct partitions, while assigning relations that include both city and country (such as `cityCapitalOfCountry`) to only one of these partitions. We refer to this partitioning of the concept graph as an *ontological partitioning*.

Given an ontological partitioning of the labels and relations, generating a partition-

ing of the extractions is straightforward. Each ontological partition specifies a cluster of related labels and relations. Using these clusters of relations and labels, we can create a corresponding partition of the extractions of specific instances of these relations and labels, where all extractions pertaining to the relations and labels in the cluster will be assigned to the same partition. Since each label and relation is assigned to a unique cluster and each extraction pertains to a single label or relations, each extraction will be assigned to exactly one partition.

The advantages of ontological partitioning directly address the weaknesses of partitioning the extraction graph directly. Partitions are based on ontological relationships between these relations and labels instead of relationships between entities. Ontological partitioning tries to maximize the number of ontological relationships preserved in the data, rather than the number of entity-based relationships. Beyond capturing important model properties in the partitioning, ontological partitioning also has a scalability advantage. The ontology is many orders of magnitude smaller than the extractions, and the size of the ontology can remain relatively constant while the number of extractions grows by many orders of magnitude. Despite these advantages, there are some critical shortcomings of this approach, which we discuss and address in the next sections.

6.3.2.1 Handling Unevenly Distributed Extractions

One issue that may arise from partitioning the concept graph is that the induced partitioning of extractions may be imbalanced. Imbalanced partitions may occur because some relations and labels occur more frequently in the data than others. For example, the ex-

tractor may have far more extractions about the label `city` than the label `bird`, resulting in more extractions in the partition induced by the cluster containing the label `city`.

Unbalanced partitions pose a problem when inference when using horizontal partitioning. Since inference is run in parallel across machines, the inference time is equal to the time taken by the slowest partition. Since inference time depends on the size of the partition, having a single large partition can result in a long inference time, even if most partitions finish quickly. By balancing partitions, we can produce the quickest overall inference.

Balancing partitions of the extraction graph directly can be accomplished by adding a constraint that all partitions contain an equal number of entities. However, applying this technique to the concept graph is problematic since, as discussed, each relation or label corresponds to a varying number of extractions. To address the problem of imbalanced partitions, we incorporate information about the data distribution into the concept graph.

Instead of treating labels and relations as atomic elements of a partitioning, we assign an importance to each label and relation based on the data distribution. The simplest way to do this is to assign each of the vertices corresponding to a label or relation a weight equal to the frequency of the label or relation in the data. When clustering the graph, we add an additional constraint that each cluster contain vertices with an equal weight. By adding such a constraint, we ensure that each cluster of the concept graph maps to an equal number of extractions. Thus, when inducing a partitioning over extractions, each partition will contain a roughly equal number of extractions.

6.3.2.2 Dealing with Unbalanced Ontologies

By creating balanced partitions, our approach ensures fast inference. However, an equally important goal is ensuring that inference quality remains high. One possible concern that may impact inference quality is the distribution of ontological information. Simply partitioning the ontology graph when the ontological information is imbalanced may lead to poor inference quality.

For example, in NELL's ontology there are nearly 50K RMUT constraints and only 418 DOM constraints. Many of the mutually-exclusive relations are not present in the extractions for the same pair of entities, while domain constraints are relevant for every extracted relation. For example, the ontological relationship RMUT(`cityCapitalOfCountry`,`currencyCountry`) which states that the relation specifying that a city is a capital of a country and the relation specifying the currency a country uses are mutually exclusive, may not be applied frequently if the extractors do not commonly confuse cities and currencies. However the ontological DOM(`cityCapitalOfCountry`, `country`) may be useful whenever the `cityCapitalOfCountry`relationship is observed.

To capture the variable utility of different ontological information, we assign a weight to each type of ontological relationships. While many different approaches can be used to select these weights, we introduce a fairly straightforward heuristic based on rarity. Specifically, we consider ontological relationships that occur rarely as more important, and assign a higher penalty to clusterings that exclude such relationships. In the perspective of a minimum cut, we can interpret this heuristic as assigning a weight to each

edge of a given type of ontological relationship.

Our approach determines weights by choosing a weight that is inversely proportional to the number of ontological constraints of that type. Using the statistics from the NELL ontology from the previous paragraph, this means that the RMUT ontological relationship would have a weight of $\frac{1}{50K}$ while the DOM ontological relationship would have a weight of $\frac{1}{418}$. Note that the aggregate weight of each type of ontological relationships is equal to one across the concept graph (e.g. there are 418 DOM edges, each with a weight of $\frac{1}{418}$). This, in some sense, can be seen as equalizing the importance across all types of ontological relationships in the concept graph.

6.4 Evaluation

While the choice of partitioning technique can influence running time, the number of partitions used in inference can also impact the computational performance of KGI. Joint inference without partitioning preserves all dependencies between extractions, but has a correspondingly complex model and cannot benefit from parallelism. Using a large number of partitions increases parallelism and improves the speed of inference, but necessarily involves losing dependencies which may reduce the quality of the inference results. We explore the speed-quality tradeoff between the number of partitions and the quality of the inference results.

6.4.1 Comparison of Partitioning Techniques

We evaluate different partitioning strategies for our KGI model with data from iteration 165 of the NELL system, which contains nearly 80K ontological relationships, 1.7M candidate facts, and 440K previously promoted facts which we represent as a separate, noisy source. While PSL supports weight learning, our experiments use a simpler setting where all source weights are set to be equal ($\forall T : w_{CL-T} = w_{CR-T} = 1$) , while entity resolution rules and ontology rules are given higher weights ($w_{ER} = w_{EL} = 10; w_O = 100$). We assess the quality of our inference results using a manually-labeled evaluation set with 4.5K extractions (Jiang et al., 2012) and measuring the running time on a 16-core Xeon X5550 CPU at 2.67GHz with 78GB of RAM. We provide documentation, code and datasets to replicate our results on GitHub.[1] To partition the ontology, we use the METIS graph partitioning package(Karypis and Kumar, 1998). In all cases, the time for partitioning the ontology graph was less than a minute. Our experiments consider two aspects of partitioning extractions for KGI: the partitioning technique and the number of partitions.

We compare four techniques for partitioning knowledge graphs with inference on the full set of extractions. The first, a `baseline`, randomly assigns each extraction to a partition. While such an approach balances partitions, it does not actively try to maintain the dependencies between extractions that KGI uses. The second approach `Onto-EqEdg-NoVtx` formulates an ontology graph where each edge (corresponding to an ontological constraint) has equal weight. The ontology graph is partitioned using a

[1] `https://github.com/linqs/KnowledgeGraphIdentification`

Table 6.1: Comparing different partitioning techniques, we find that partitioning extractions with an ontology-based approach that weights vertices with the frequency of respective labels and relations in the data preserves model quality and reduces inference speed

Technique	AUC	Opt. Terms	Time (min.)
NELL (no KGI)	0.765	-	-
baseline	0.780	3.0M	**31**
Onto-EqEdg-NoVtx	0.788	4.2M	42
Onto-EqEdg-WtVtx	**0.791**	3.7M	**31**
Onto-WtEdg-WtVtx	0.790	3.7M	31
No-Partitioning	0.794	10.9M	97

p-way balanced min-cut, where the objective function minimizes the communication cost defined by the sum of adjacent edge weights. The third approach, Onto-EqEdg-WtVtx equally weights each edge but assigns weights to each vertex (relation or label) based on the frequency of that relation or label in the extraction data. The ontology graph is partitioned using a minimum edge cut with a constraint that each cluster has the same aggregate vertex weight. The fourth approach, Onto-WtEdg-WtVtx weights vertices by frequency, and also assigns a weight to each edge. The edge weights are set to be inversely proportional to the frequency of the respective type of ontological information. This formulation was chosen to give each type of ontological information an equal representation in the ontology graph.

For each partitioning algorithm, we generate 6 disjoint clusters of labels and relations. We use these 6 clusters of labels and relations to produce 6 corresponding partitions from the extraction data, where each partition is limited to the relations and labels in the corresponding cluster. For each of the 6 partitions generated we perform inference on each partition independently and combine the results of this distributed inference, aver-

aging truth values when the same fact appears in the output of multiple partitions. We compute the area under the precision-recall curve (AUC) for each technique, and report the running time of the slowest partition.

As shown in Table 6.1, inference over the full knowledge graph `No-Partitioning` takes 97 minutes and provides an improvement over NELL's default strategy for promoting candidates. The `baseline` strategy of randomly partitioning extractions dramatically reduces inference time, but produces a considerable drop in the AUC. The partitioning also fails to preserve many of the ontological relationships, with the largest partition containing only 3M ground terms. By using an ontology-aware partitioning method `Onto-EqEdg-NoVtx`, we improve the AUC over the baseline, achieving parity with the full joint inference problem, but the running time increases significantly relative to the baseline. This increase in running time can be partially explained by the increased number of optimization terms, with 4.2M terms in the largest partition. Using vertex weights that reflect the data distribution, `Onto-EqEdg-WtVtx` reduces the inference time by improving the partition balance (3.7M terms in the largest partition) while also improving AUC. However including weights based on the ontological frequency, `Onto-WtEdg-WtVtx`, does not improve our results.

6.4.2 Assessing the Impact of Partition Size

Next, we examine how the number of partitions impact the speed and quality of knowledge graph identification. We used the best-performing partitioning technique from the earlier experiments, `Onto-EqEdg-WtVtx`, to create a varying number of partitions. We

Table 6.2: Increasing the number of partitions used in inference can dramatically reduce the time for inference with relatively modest loss in quality, as measured by AUC

Partitions	AUC	Time (min.)
48	0.788	12
24	0.790	20
12	0.791	26
6	0.791	31
3	0.794	44
2	0.794	57
1	0.794	97

generated 1, 2, 3, 6, 12, 24 and 48 partitions of the extractions using this technique. We report the results for each of these partition numbers in Table 6.2. Our results show that partitioning can dramatically reduce inference time from 97 minutes with a single partition to 12 minutes with 48 partitions. Surprisingly, there is little degradation in inference quality as measured by AUC, which ranges from .794 with a single partition to .788 with 48 partitions. The quality for 48 partitions even remains higher than the baseline strategy from the previous section, which had an AUC of .780 when randomly partitioning the data into 6 partitions. Fig. 6.3 clearly shows this trade-off between inference speed and quality for the NELL dataset. The sublinear speedup, which diverges from the earlier results showing linear scaling with extraction size, is potentially related to caching effects or computational overhead in the implementation of our model.

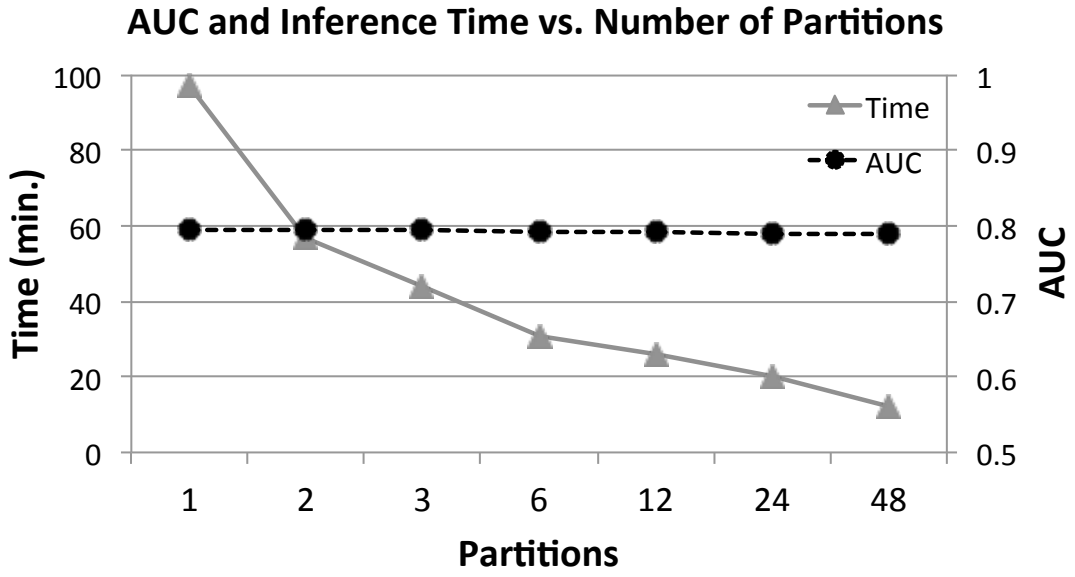


Figure 6.3: A plot showing the scaling performance of distributed KGI as a function of the number of partitions. The left axis shows the running time of the slowest partition while the right axis shows the AUPRC of the inference output. As the number of partitions increases, the inference running time decreases but the AUC fairly stable. These results indicate that partitioning KGI can improve scalability without sacrificing quality.

6.5 Discussion

In this chapter, we present a deeper examination into one of the central problems in knowledge graph construction: scalability. Our contributions are: (1) a formal analysis of the complexity of knowledge graph identification, signaling the difficulty of alternative implementations KGI; (2) a demonstration of the scalability of our KGI implementation, in experimental evaluation of several real-world knowledge graph construction tasks and through an empirical analysis of over 1000 KGI problems; (3) further improving the scalability of KGI by using a distributed approach which partitions extractions while taking into account the data distribute and preserving key ontological relationships, including

weighting these relationships by their relative frequency; and (4) performing an empirical analysis comparing different distributed approaches to KGI and exploring the tradeoff between speed and quality based on the number and size of partitions. As the results of this investigation demonstrate, despite tackling a theoretically intractable problem, KGI can be scaled to yield an order-of-magnitude decrease in inference time with little loss of quality. This scalability is a necessity for commercial settings where large problems can easily be distributed over hundreds or thousands of machines.

Chapter 7: Online Collective Inference

A key challenge of many artificial intelligence problems is that the evidence grows and changes over time, requiring updates to inferences. Every time a user rates a new movie on Netflix, posts a status update on Twitter, or adds a connection on LinkedIn, inferences about preferences, events, or relationships must be updated. When constructing a knowledge base, each newly acquired document prompts the system to update inferences over related facts and resolve mentions to their canonical entities. Problems such as these benefit from collective (i.e., joint) reasoning, but incorporating new evidence into a collective model is particularly challenging. New evidence can affect multiple predictions, so updating inference typically involves recomputing all predictions in an expensive global optimization. Even when a full inference update is tractable—which, using the best known methods, can be linear in the number of factors—it may still be impractical. For example, updating a knowledge graph with millions of facts can take hours, thereby requiring some compromise, either in the form of a deferment strategy or approximate update. In this chapter, we consider the task of efficiently updating the *maximum-a-posteriori* (MAP) state of a probabilistic graphical model, conditioned on evolving evidence. We refer to this problem as *online collective inference*.

In online collective inference, a single graphical model, describing the conditional

distribution of a set of random variables with fixed dependency structure, is given. Over a series of epochs, the true assignments (i.e., labels) of certain variables are revealed, introducing new evidence with which can be used to update the assignments to the remaining unknowns. We constrain the problem by adding a budget, such that only a fixed percentage of variables can be updated in each epoch. The introduction of a budget necessitates some approximation to full inference. This constraint distinguishes our work from the vast body of literature on belief revision (e.g., Gardenfors, 1992), Bayesian network updates (e.g., Buntine, 1991; Friedman and Goldszmidt, 1997; Li et al., 2006), models for dynamic (Murphy, 2002) or sequential (Fine et al., 1998) data, and adaptive inference (e.g., Acar et al., 2008), which deal with exact updates to inference. We analyze budgeted online collective inference from both the theoretical and algorithmic perspectives, addressing two fundamental questions: How do we choose which variables to update? How "close" is the approximate inference update to the full inference update?

To formalize the latter question, we introduce the concept of *inference regret*. Informally, inference regret measures the amount of change induced by fixing (i.e., conditioning on) certain variables in the inference optimization. We specifically analyze the inference regret of continuous graphical models whose inference objective is strongly convex. One instantiation of this class of models is hinge-loss Markov random fields (Bach et al., 2015), which have been used throughout this dissertation, and have been broadly applied and demonstrate state-of-the-art performance in many applications. Using the duality between strong convexity and stability, we upper-bound the inference regret. Our bound is proportional to the distance from the fixed variables to the optimal values of the full inference problem, scaled by a function of several model-specific properties. We use the

inference regret bound to quantify the effect of approximate inference updates in response to new evidence (in this case, revealed labels). The bound highlights two terms affecting the regret: the accuracy of the original predictions and the amount that the original predictions change. This latter insight informs the design of approximate update methods with a simple intuition: fix the predictions that are unlikely to change in a full inference update.

To efficiently determine which variables are least likely to change, we turn to the optimization algorithm used during inference. The alternating direction method of multipliers (ADMM) (Boyd et al., 2011) is a popular convex optimization technique that offers convergence guarantees while remaining highly scalable. We analyze the optimization process and catalog the features that allow us to determine which variables will change the most. Using these features to generate a score for each variable, we establish a ranking capturing the priority of including the variables in subsequent inference. Since the variable scores are produced using the state of the optimization algorithm, our method does not incur computational overhead. By ranking variables, we approximate full inference with an arbitrary budget and support an anytime online inference algorithm.

7.1 Preliminaries

The theory and methods introduced in this chapter apply to any continuous-valued MRF with a strongly convex MAP inference objective function. One case of particular interest is *hinge-loss Markov random fields* (HL-MRFs). An HL-MRF is a continuous-valued Markov network in which the potentials are hinge functions of the variables. Our

choice of HL-MRFs comes from technical considerations: we reason about the strength of convexity of the inference objective, and *maximum a posteriori* (MAP) inference in HL-MRFs can be strongly convex.

To better understand HL-MRFs and PSL, consider a model for collective classification of network data, in which the goal is to assign labels to nodes, conditioned on some local evidence and network structure. Let $G \triangleq (\mathcal{V}, \mathcal{E})$ denote an undirected graph on $n \triangleq |\mathcal{V}|$ nodes. Each node $i \in \mathcal{V}$ is associated with a set of local observations, X_i, and an unknown label, L_i. (In some settings, a subset of the labels are revealed.) In general, the observations and labels can be real-valued; but for simplicity of exposition, let us assume that each observation is binary-valued, and each label is categorical. The following logical rules define a PSL program for a typical collective classification model:

$$w_{x,\ell} : \textsc{Feature}(N, x) \Rightarrow \textsc{Label}(N, \ell)$$

$$w_{e,\ell} : \textsc{Edge}(N_1, N_2) \wedge \textsc{Label}(N_1, \ell) \Rightarrow \textsc{Label}(N_2, \ell)$$

Variables N, N_1 and N_2 denote nodes; x indexes a local feature; and ℓ denotes a label. The rules are weighted by $w_{x,\ell}$ and $w_{e,\ell}$ respectively. Given G and $\mathbf{X} \triangleq (X_1, \ldots, X_n)$ (and possibly some subset of the labels), the rules are *grounded out* for all possible instantiations of the predicates. The groundings involving unknown variables—in this case, groundings of the LABEL predicate—are represented by $[0, 1]$-valued variables, $\mathbf{Y} \triangleq (Y_{i,\ell})_{i,\ell}$. Using a relaxation of the MAX-SAT problem to continuous domains (Globerson and Jaakkola, 2007), each grounding is converted to a convex hinge function

of the form

$$f(\mathbf{X}, \mathbf{Y}) = (\max\{0, \varphi(\mathbf{X}, \mathbf{Y})\})^q ,$$

where φ is a linear function of $(\mathbf{X}, \mathbf{Y})$, and $q \in \{1, 2\}$ is an exponent that is set *a priori* for the given rule. Each hinge function becomes a potential in an HL-MRF.

The resulting HL-MRF enables probabilistic inference over the set of PSL rules. Fix a set of r PSL rules, with corresponding weights $\mathbf{w} \triangleq (w_1, \ldots, w_r)$. For the i^{th} rule, let $\mathcal{G}(i)$ denote its set of groundings in G, and let f_j^i denote the j^{th} grounding of its associated hinge function. To compactly express the weighted sum of grounded rules, we let

$$\mathbf{f}(\mathbf{X}, \mathbf{Y}) \triangleq \left[\sum_{j=1}^{|\mathcal{G}(1)|} f_j^1(\mathbf{X}, \mathbf{Y}) , \ \ldots , \ \sum_{j=1}^{|\mathcal{G}(r)|} f_j^r(\mathbf{X}, \mathbf{Y}) \right]^\top$$

denote the aggregate of the grounded hinge functions. We can thus write the weighted sum of groundings as $\mathbf{w} \cdot \mathbf{f}(\mathbf{X}, \mathbf{Y})$. This inner product defines a distribution over $(\mathbf{Y} \mid \mathbf{X})$ with probability density function $p(\mathbf{Y} = \mathbf{y} \mid \mathbf{X} = \mathbf{x}; \mathbf{w}) \propto \exp(-\mathbf{w} \cdot \mathbf{f}(\mathbf{X}, \mathbf{Y}))$. The maximizer of the density function (alternately, the minimizer of $-\mathbf{w} \cdot \mathbf{f}(\mathbf{X}, \mathbf{Y})$) is the MAP state. The values of $\mathbf{Y}$ in the MAP state can be interpreted as confidences. Additionally, we can define a prior distribution over each $\mathbf{Y}$. In this case, we will use an L^2, or Gaussian, prior. This can be accomplished using the rule $w_{p,\ell} : \neg \textsc{Label}(N, \ell)$, with a squared hinge (i.e., $q = 2$). Let us assume, without loss of generality, that each prior rule has weight $w_{p,\ell} = w_p/2$, for some $w_p > 0$. Thus, the corresponding hinge function for grounding $\textsc{Label}(i, \ell)$ is simply $(Y_{i,\ell})^2$; the aggregate features for the prior are $\|\mathbf{Y}\|_2^2$. So

as to simplify notation, let $\dot{\mathbf{w}} \triangleq (\mathbf{w}, w_p)$ and define an *energy* function,

$$E(\mathbf{y} \,|\, \mathbf{x}; \dot{\mathbf{w}}) \triangleq \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) + \frac{w_p}{2} \|\mathbf{y}\|_2^2 . \tag{7.1}$$

The resulting probability density function is

$$p\left(\mathbf{Y} = \mathbf{y} \,|\, \mathbf{X} = \mathbf{x}; \dot{\mathbf{w}}\right) \propto \exp\left(-E(\mathbf{y} \,|\, \mathbf{x}; \dot{\mathbf{w}})\right).$$

MAP inference, henceforth denoted $h(\mathbf{x}; \dot{\mathbf{w}})$, is given by

$$h(\mathbf{x}; \dot{\mathbf{w}}) = \arg\min_{\mathbf{y}} \; E(\mathbf{y} \,|\, \mathbf{x}; \dot{\mathbf{w}}).$$

7.2 Inference Regret

The notion of *regret* has often been used to measure the loss incurred by an online learning algorithm relative to the optimal hypothesis. We extend this concept to online *inference*. Fix a model. Suppose we are given evidence, $\mathbf{X} = \mathbf{x}$, from which we make a prediction, $\mathbf{Y} = \mathbf{y}$, using MAP inference. Then, some subset of the unknowns are revealed. Conditioning on the new evidence, we have two choices: we can recompute the MAP state of the remaining variables, using full inference; or, we can fix some of the previous predictions, and only update a certain subset of the variables. To understand the consequences of fixing our previous predictions we must answer a basic question: how much have the old predictions changed?

We formalize the above question in the following concept.

Definition 1. Fix a budget $m \geq 1$. For some subset $\mathcal{S} \subset \{1, \ldots, n\}$, such that its complement $\overline{\mathcal{S}} \triangleq \{1, \ldots, n\} \setminus \mathcal{S}$, has size $\left|\overline{\mathcal{S}}\right| = m$, let $\mathbf{Y}_{\mathcal{S}}$ denote the corresponding subset of the variables, and let $\mathbf{Y}_{\overline{\mathcal{S}}}$ denote its complement. Assume there is an operator Γ that concatenates $\mathbf{Y}_{\mathcal{S}}$ and $\mathbf{Y}_{\overline{\mathcal{S}}}$ in the correct order. Fix a model, $\dot{\mathbf{w}}$, and an observation, $\mathbf{X} = \mathbf{x}$. Further, fix an assignment, $\mathbf{Y}_{\mathcal{S}} = \mathbf{y}_{\mathcal{S}}$, and let

$$h(\mathbf{x}, \mathbf{y}_{\mathcal{S}}; \dot{\mathbf{w}}) \triangleq \Gamma\left(\mathbf{y}_{\mathcal{S}}, \operatorname*{arg\,min}_{\mathbf{y}_{\overline{\mathcal{S}}}} E\left(\Gamma(\mathbf{y}_{\mathcal{S}}, \mathbf{y}_{\overline{\mathcal{S}}}) \mid \mathbf{x}; \dot{\mathbf{w}}\right)\right)$$

denote the new MAP configuration for $\mathbf{Y}_{\overline{\mathcal{S}}}$ after fixing $\mathbf{Y}_{\mathcal{S}}$ to $\mathbf{y}_{\mathcal{S}}$. We define the *inference regret* for $(\mathbf{x}, \mathbf{y}_{\mathcal{S}}; \dot{\mathbf{w}})$ as

$$\mathfrak{R}_n(\mathbf{x}, \mathbf{y}_{\mathcal{S}}; \dot{\mathbf{w}}) \triangleq \frac{1}{n} \left\| h(\mathbf{x}; \dot{\mathbf{w}}) - h(\mathbf{x}, \mathbf{y}_{\mathcal{S}}; \dot{\mathbf{w}}) \right\|_1 . \tag{7.2}$$

In general, the inference regret can be as high as 1 for variables in [0,1]. For example, consider network classification model in which probability mass is only assigned to configurations where all nodes have the same label. Fixing a variable corresponding to a single node label in this setting is tantamount to fixing the label for all nodes. In the presence of strong evidence for a different label, incorrectly fixing a single variable results in incorrectly inferring all variables.

In online inference, regret can come from two sources. First, there is the regret of not updating the MAP state given new evidence (in this case, revealed labels). If this regret is low, it may not be worthwhile to update inference, which can be useful in situations where updating inference is expensive (such as updating predicted attributes

for all users in a social network). The second type of regret is from using an approximate inference update in which only certain variables are updated, while the rest are kept fixed to their previous values. We describe several such approximations in Section 7.3. In practice, one may have both types of regret, caused by approximate updates in response to new evidence. Note that the inference regret obeys the triangle inequality, so one can upper-bound the compound regret of multiple updates using the regret of each update.

7.2.1 Regret Bounds for Strongly Convex Inference

A convenient property of the L^2 prior is that it is *strongly convex*, by which we mean the following.

Definition 2. Let $\Omega \subseteq \mathbb{R}^n$ denote a convex set. A differentiable function, $f : \Omega \to \mathbb{R}$, is κ-*strongly convex* (w.r.t. the 2-norm) if, for all $\boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega$,

$$\frac{\kappa}{2} \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 + \langle \nabla f(\boldsymbol{\omega}), \boldsymbol{\omega}' - \boldsymbol{\omega} \rangle \leq f(\boldsymbol{\omega}') - f(\boldsymbol{\omega}). \tag{7.3}$$

Strong convexity has a well-known duality with stability (Wainwright, 2006), which we will use in our theoretical analysis.

The function $f(\boldsymbol{\omega}) \triangleq \frac{1}{2} \|\boldsymbol{\omega}\|_2^2$ is 1-strongly convex. Therefore, the prior, $\frac{w_p}{2} \|\mathbf{y}\|_2^2$, is at least w_p-strongly convex. We also have that the aggregated hinge functions, $\mathbf{f}(\mathbf{x}, \mathbf{y})$, are convex functions of $\mathbf{Y}$. Thus, it is easily verified that the energy, $E(\mathbf{y} \,|\, \mathbf{x}; \dot{\mathbf{w}})$, is at least a w_p-strongly convex function of $\mathbf{y}$. This yields the following upper bound on the inference regret.

Proposition 1. *Fix a model with weights* $\dot{\mathbf{w}}$*. Assume there exists a constant* $B \in [0, \infty)$ *such that, for any* $\mathbf{x}$*, and any* $\mathbf{y}, \mathbf{y}'$ *that differ at coordinate* i*,*

$$\|\mathbf{f}(\mathbf{x}, \mathbf{y}) - \mathbf{f}(\mathbf{x}, \mathbf{y}')\|_2 \leq B \left| y_i - y_i' \right|. \tag{7.4}$$

Then, for any observations $\mathbf{x}$*, any budget* $m \geq 1$*, any subset* $\mathcal{S} \subset \{1, \ldots, n\} : \left|\overline{\mathcal{S}}\right| = m$*, and any assignments* $\mathbf{y}_{\mathcal{S}}$*, with* $\hat{\mathbf{y}} \triangleq h(\mathbf{x}; \dot{\mathbf{w}})$*, we have that*

$$\mathfrak{R}_n(\mathbf{x}, \mathbf{y}_{\mathcal{S}}; \dot{\mathbf{w}}) \leq \sqrt{\frac{1}{n}\left(\frac{3}{2} + \frac{B\,\|\mathbf{w}\|_2}{w_p}\right)\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\|_1}.$$

Proof

Let $\hat{\mathbf{y}} \triangleq h(\mathbf{x}; \dot{\mathbf{w}})$ denote the original MAP configuration, i.e., the minimizer of $E(\cdot \mid \mathbf{x}; \mathbf{w})$. Let $\hat{\mathbf{y}}' \triangleq h(\mathbf{x}, \mathbf{y}_{\mathcal{S}}; \dot{\mathbf{w}})$ denote the updated MAP state after conditioning, and note that $\hat{\mathbf{y}}'_{\overline{\mathcal{S}}}$ is the minimizer of $E\left(\Gamma(\mathbf{y}_{\mathcal{S}}, \cdot) \mid \mathbf{x}; \dot{\mathbf{w}}\right)$.

Since $\hat{\mathbf{y}}_{\mathcal{S}}$ may be different from $\mathbf{y}_{\mathcal{S}}$, we have that $\hat{\mathbf{y}}$ may not be in the domain of $E\left(\Gamma(\mathbf{y}_{\mathcal{S}}, \cdot) \mid \mathbf{x}; \dot{\mathbf{w}}\right)$.

We therefore define a vector $\tilde{\mathbf{y}} \in [0, 1]^n$ that *is* in the domain, and has minimal Hamming distance to $\hat{\mathbf{y}}$. Let $\tilde{y}_i \triangleq y_i$ for all $i \in \mathcal{S}$, and $\tilde{y}_j \triangleq \hat{y}_j$ for all $j \notin \mathcal{S}$.

We begin by restating the two-norm in terms of $\tilde{\mathbf{y}}$,

$$\begin{aligned}
&\left\|\hat{\mathbf{y}}' - \hat{\mathbf{y}}\right\|_2^2 \\
&\quad = \left\|\hat{\mathbf{y}}' - \tilde{\mathbf{y}} + \tilde{\mathbf{y}} - \hat{\mathbf{y}}\right\|_2^2. \\
&\quad = \left\|\hat{\mathbf{y}}' - \tilde{\mathbf{y}}\right\|_2^2 + \left\|\tilde{\mathbf{y}} - \hat{\mathbf{y}}\right\|_2^2 + 2(\hat{\mathbf{y}}' - \tilde{\mathbf{y}})(\tilde{\mathbf{y}} - \hat{\mathbf{y}})
\end{aligned}$$

However, by the construction of $\tilde{\mathbf{y}}$, each element of the vector equals either $\hat{\mathbf{y}}'$ or $\hat{\mathbf{y}}$ so the final dot product will yield 0. This gives us

$$\left\|\hat{\mathbf{y}}' - \hat{\mathbf{y}}\right\|_2^2 = \left\|\hat{\mathbf{y}}' - \tilde{\mathbf{y}}\right\|_2^2 + \left\|\tilde{\mathbf{y}} - \hat{\mathbf{y}}\right\|_2^2. \tag{7.5}$$

Further, since the domain of each Y_i is $[0, 1]$, the L^1 distance dominates the L^2 distance.

$$\left\|\tilde{\mathbf{y}} - \hat{\mathbf{y}}\right\|_2^2 = \left\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\right\|_2^2 \leq \left\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\right\|_1. \tag{7.6}$$

Therefore, combining Equations 7.5 and 7.6,

$$\begin{aligned}
&\left\|\hat{\mathbf{y}} - \hat{\mathbf{y}}'\right\|_2^2 \\
&\quad = \frac{1}{2}\left(\left\|\hat{\mathbf{y}} - \hat{\mathbf{y}}'\right\|_2^2 + \left\|\hat{\mathbf{y}}' - \hat{\mathbf{y}}\right\|_2^2\right) \\
&\quad \leq \frac{1}{2}\left(\left\|\hat{\mathbf{y}} - \hat{\mathbf{y}}'\right\|_2^2 + \left\|\hat{\mathbf{y}}' - \tilde{\mathbf{y}}\right\|_2^2 + \left\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\right\|_1\right).
\end{aligned} \tag{7.7}$$

For any κ-strongly convex function, $\varphi : \Omega \to \mathbb{R}$, where $\hat{\boldsymbol{\omega}} = \arg\min_{\boldsymbol{\omega} \in \Omega} \varphi(\boldsymbol{\omega})$ is

the minimizer, then $\forall\, \boldsymbol{\omega}' \in \Omega$,

$$\frac{1}{2} \left\| \hat{\boldsymbol{\omega}} - \boldsymbol{\omega}' \right\|_2^2 \leq \frac{1}{\kappa} \left(\varphi(\boldsymbol{\omega}') - \varphi(\hat{\boldsymbol{\omega}}) \right). \tag{7.8}$$

Applying this identify to the first two terms in Equation 7.7, since $E(\cdot \,|\, \mathbf{x}; \dot{\mathbf{w}})$ is w_p-strongly convex, we have that

$$\begin{aligned}
&\frac{1}{2} \left\| \hat{\mathbf{y}} - \hat{\mathbf{y}}' \right\|_2^2 + \frac{1}{2} \left\| \hat{\mathbf{y}}' - \tilde{\mathbf{y}} \right\|_2^2 \\
&\qquad \leq \quad \frac{1}{w_p} \left(E(\hat{\mathbf{y}}' \,|\, \mathbf{x}; \dot{\mathbf{w}}) - E(\hat{\mathbf{y}} \,|\, \mathbf{x}; \dot{\mathbf{w}}) \right) \\
&\qquad\quad + \frac{1}{w_p} \left(E(\tilde{\mathbf{y}} \,|\, \mathbf{x}; \dot{\mathbf{w}}) - E(\hat{\mathbf{y}}' \,|\, \mathbf{x}; \dot{\mathbf{w}}) \right) \\
&\qquad \leq \frac{1}{w_p} \left(E(\tilde{\mathbf{y}} \,|\, \mathbf{x}; \dot{\mathbf{w}}) - E(\hat{\mathbf{y}} \,|\, \mathbf{x}; \dot{\mathbf{w}}) \right).
\end{aligned} \tag{7.9}$$

The $E(\hat{\mathbf{y}}' \,|\, \mathbf{x}; \dot{\mathbf{w}})$ terms cancel out. Expanding $E(\cdot \,|\, \mathbf{x}; \dot{\mathbf{w}})$,

$$\begin{aligned}
&E(\tilde{\mathbf{y}} \,|\, \mathbf{x}; \dot{\mathbf{w}}) - E(\hat{\mathbf{y}} \,|\, \mathbf{x}; \dot{\mathbf{w}}) \\
&= \mathbf{w} \cdot (\mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}) - \mathbf{f}(\mathbf{x}, \hat{\mathbf{y}})) + \frac{w_p}{2} \left(\|\tilde{\mathbf{y}}\|_2^2 - \|\hat{\mathbf{y}}\|_2^2 \right) \\
&\leq \|\mathbf{w}\|_2 \left\| \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}) - \mathbf{f}(\mathbf{x}, \hat{\mathbf{y}}) \right\|_2 + \frac{w_p}{2} \left(\|\tilde{\mathbf{y}}\|_2^2 - \|\hat{\mathbf{y}}\|_2^2 \right) \\
&\leq \|\mathbf{w}\|_2 \left\| \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}) - \mathbf{f}(\mathbf{x}, \hat{\mathbf{y}}) \right\|_2 + w_p \left\| \mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}} \right\|_1.
\end{aligned} \tag{7.10}$$

The first inequality uses Cauchy-Schwarz and the final step uses

$$\|\tilde{\mathbf{y}}\|_2^2 - \|\hat{\mathbf{y}}\|_2^2 \leq 2 \left\| \mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}} \right\|_1.$$

Finally, we construct a series of vectors, indexed by each $i \in \mathcal{S}$, that transform $\hat{\mathbf{y}}$ into $\tilde{\mathbf{y}}$, one coordinate at a time. For the following, let $\mathcal{S}(j)$ denote the j^{th} element in $\mathcal{S}$. First, let $\tilde{\mathbf{y}}^{(0)} \triangleq \hat{\mathbf{y}}$; then, for $j = 1, \ldots, m$, let $\tilde{\mathbf{y}}^{(j)}$ be equal to $\tilde{\mathbf{y}}^{(j-1)}$ with index $\mathcal{S}(j)$ replaced with value $\tilde{y}_{\mathcal{S}(j)}$. Note that $\tilde{\mathbf{y}}^{(m)} = \tilde{\mathbf{y}}$. Using the triangle inequality, one can show that

$$\begin{aligned}
\|\mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}) - \mathbf{f}(\mathbf{x}, \hat{\mathbf{y}})\|_2 &= \left\|\mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}^{(m)}) - \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}^{(0)})\right\|_2 \\
&= \left\|\sum_{j=1}^{m} \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}^{(j)}) - \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}^{(j-1)})\right\|_2 \\
&\leq \sum_{j=1}^{m} \left\|\mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}^{(j)}) - \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}^{(j-1)})\right\|_2 \\
&\leq B \left\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\right\|_1 . \qquad (7.11)
\end{aligned}$$

The last inequality uses Equation 7.4, since $\tilde{\mathbf{y}}^{(j)}$ and $\tilde{\mathbf{y}}^{(j-1)}$ differ at a single coordinate, $\mathcal{S}(j)$.

Combining Equations 7.7, 7.9 and 7.10 we have

$$\begin{aligned}
&\left\|\hat{\mathbf{y}} - \hat{\mathbf{y}}'\right\|_2^2 \\
&\quad\leq \frac{1}{2}\left(\left\|\hat{\mathbf{y}} - \hat{\mathbf{y}}'\right\|_2^2 + \left\|\hat{\mathbf{y}}' - \tilde{\mathbf{y}}\right\|_2^2 + \left\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\right\|_1\right) \\
&\quad\leq \frac{1}{w_p}\left(\|\mathbf{w}\|_2 \left\|\mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}) - \mathbf{f}(\mathbf{x}, \hat{\mathbf{y}})\right\|_2 + w_p \left\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\right\|_1\right) + \frac{1}{2}\left\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\right\|_1 \\
&\quad\leq \frac{1}{w_p}\|\mathbf{w}\|_2 \left\|\mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}) - \mathbf{f}(\mathbf{x}, \hat{\mathbf{y}})\right\|_2 + \frac{3}{2}\left\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\right\|_1 .
\end{aligned}$$

Using the result from Equation 7.11, we get

$$\begin{aligned}
&\left\|\hat{\mathbf{y}} - \hat{\mathbf{y}}'\right\|_2^2 \\
&\quad \leq \frac{1}{w_p} \|\mathbf{w}\|_2 B \|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\|_1 + \frac{3}{2} \|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\|_1 \\
&\quad \leq \left(\frac{3}{2} + \frac{B \|\mathbf{w}\|_2}{w_p}\right) \|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\|_1 \,. \qquad (7.12)
\end{aligned}$$

We then multiply both sides of the inequality by $1/n$ and take the square root. Using $\frac{1}{n}\left\|\hat{\mathbf{y}} - \hat{\mathbf{y}}'\right\|_1 \leq \frac{1}{\sqrt{n}}\left\|\hat{\mathbf{y}} - \hat{\mathbf{y}}'\right\|_2$ finishes the proof.

$$\frac{1}{\sqrt{n}}\left\|\hat{\mathbf{y}} - \hat{\mathbf{y}}'\right\|_2 \leq \frac{1}{\sqrt{n}}\sqrt{\left(\frac{3}{2} + \frac{B \|\mathbf{w}\|_2}{w_p}\right) \|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\|_1} \,.$$

■

Proposition 1 states that the inference regret is proportional to the L^1 distance from $\mathbf{y}_{\mathcal{S}}$ to $\hat{\mathbf{y}}_{\mathcal{S}}$, multiplied by a model-dependent quantity, $\mathrm{O}\left(\frac{B\|\mathbf{w}\|_2}{n\, w_p}\right)$. Later in this section, we discuss how to bound the features' Lipschitz constant, B, demonstrating that it is typically a small constant (e.g., 1). Thus, assuming $\|\mathbf{w}\|_2$ is bounded from above, and the weight on the prior, w_p, is bounded from below, the model-dependent term should decrease with the number of variables, n. For variables bounded in $[0, 1]$, the Hamming distance upper-bounds the L^1 distance. Using this identity, a pessimistic upper bound for the distance term is $\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\|_1 \leq |\mathcal{S}|$. In this case, the regret is proportional to $\mathrm{O}\big(\sqrt{|\mathcal{S}|/n}\big)$; i.e., the square root of the fraction of the variables that are fixed. While this yields a uniform, analytic upper bound, we gain more insight by considering the specific contexts.

For instance, suppose $\mathbf{y}_{\mathcal{S}}$ is a set of labels that has been revealed. Then $\mathfrak{R}_n(\mathbf{x}, \mathbf{y}_{\mathcal{S}}; \dot{\mathbf{w}})$ is the regret of not updating inference conditioned on new evidence, and $\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\|_1$ is the L^1 error of the original predictions w.r.t. the true labels. Now, suppose $\mathbf{y}_{\mathcal{S}}$ is a set of labels that are fixed from a previous round of inference. Then $\mathfrak{R}_n(\mathbf{x}, \mathbf{y}_{\mathcal{S}}; \dot{\mathbf{w}})$ is the regret of an approximate inference update, and $\|\mathbf{y}_{\mathcal{S}} - \hat{\mathbf{y}}_{\mathcal{S}}\|_1$ is the L^1 distance between the old predictions and the new predictions in the full inference update. Thus, to minimize this regret, we must fix values that are already close to what we think they will be in the updated MAP state. This criteria motivates our approximate update methods in Section 7.3.

7.2.2 The Lipschitz Constant of the Features

In this section, we give some intuition on how to bound the Lipschitz constant of the features, B, by considering a specific example. Suppose the model has a single rule: $X \Rightarrow Y$. The corresponding hinge is $f(X, Y) \triangleq \max\{0, X - Y\}$. Using the fact that $|\max\{0, a\} - \max\{0, b\}| \leq |a - b|$, one can show that $\|\mathbf{f}(\mathbf{x}, \mathbf{y}) - \mathbf{f}(\mathbf{x}, \mathbf{y}')\|_2 \leq |y_i - y_i'| \leq 1$, so $B = 1$.

PSL models typically use rules of this nature, with varying arity (i.e., diadic, triadic, etc.). In general, B should grow linearly with the number of groundings involving any single variable (i.e., the maximum degree of the factor graph). The number of groundings generated by each rule depends on its arity and the data. For instance, the relational rule in Section 7.1 will ground out once for each edge and each label; if there are 2 labels, and the maximum degree is bounded by a constant, Δ, then the number of groundings generated by this rule for any single variable is at most 2Δ. Thus, in many practical models, B will

be a small constant.

7.3 Algorithms for Online Inference Activation

The bounds presented in Section 7.2.1 suggest that online collective inference under budget constraints is close to the full inference update when one is able to successfully choose and fix variables whose inferred values will have little or no change. We refer to the complementary process of selecting which variables to infer as *activation*. In practice, designing an activation algorithm is difficult. The optimization problem required to choose a set of variables, each with heterogeneous regret and optimization cost, that do not exceed an optimization budget is an instance of the NP-hard knapsack problem. Given the intrinsic intractability of selecting an optimal set of variables, we present two algorithms that employ theoretical insights from the previous section and show promise in empirical experiments.

7.3.1 Background: ADMM Optimization

To develop activation algorithms, we turn to the optimization technique used to determine the MAP state in HL-MRFs. Bach et al. (2012) have shown that applying consensus optimization using the Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011) provides scalable inference for HL-MRFs. For clearer exposition, we express the inference in terms of the set of ground rules, $\mathcal{G}$ and rewrite the energy function in Section 7.1 as:

$$E(\mathbf{y} \mid \mathbf{x}; \dot{\mathbf{w}}) \triangleq \sum_{g \in \mathcal{G}} w_g f_g(\mathbf{x}, \mathbf{y}) + \frac{w_p}{2} \left\| \mathbf{y} \right\|_2^2$$

Here, $w_g f_g(\mathbf{x}, \mathbf{y})$ is a weighted potential corresponding to a single ground rule. ADMM substitutes the global optimization problem with local optimizations for each potential using independent copies of the variables. For each grounding $g \in \mathcal{G}$, let $\mathbf{y}_g$ denote the variables involved in g and $\tilde{\mathbf{y}}_g$ indicate the local copy of those variables. To reconcile the local optimizations, ADMM introduces a constraint that local variable copies agree with the global "consensus" for each variable i involved in the grounding; that is, $\mathbf{y}_g[i] = \tilde{\mathbf{y}}_g[i]$. This constraint is transformed into an augmented Lagrangian with penalty parameter $\rho > 0$ and Lagrange multipliers $\boldsymbol{\alpha}_g$:

$$\min_{\tilde{\mathbf{y}}_g} \; w_g \, f_g(\mathbf{x}, \tilde{\mathbf{y}}_g) + \frac{\rho}{2} \left\| \tilde{\mathbf{y}}_g - \mathbf{y}_g + \frac{1}{\rho} \boldsymbol{\alpha}_g \right\|^2 \tag{7.13}$$

ADMM iteratively alternates optimizing the local potentials, then updating the consensus estimates and associated Lagrange multipliers for each variable, as such:

$$\tilde{\mathbf{y}}_g \leftarrow \text{argmin}_{\tilde{\mathbf{y}}_g} w_g \, f_g(\mathbf{x}, \tilde{\mathbf{y}}_g) + \frac{\rho}{2} \left\| \tilde{\mathbf{y}}_g - \mathbf{y}_g + \frac{1}{\rho} \boldsymbol{\alpha}_g \right\|^2 ;$$

$$\mathbf{y}[i] \leftarrow \text{mean}_g(\tilde{\mathbf{y}}_g[i]) \; ; \;\; \boldsymbol{\alpha}_g[i] \leftarrow \boldsymbol{\alpha}_g[i] + \rho(\tilde{\mathbf{y}}_g[i] - \mathbf{y}_g[i]) \, .$$

A key element of this optimization is the interplay of two components: the weighted potential corresponding to a grounding and the Lagrangian penalty for deviating from the consensus estimate. As optimization proceeds, the Lagrange multipliers are updated to increase the penalty for deviating from the global consensus. At convergence, a balance exists between the two components, reconciling the local minimizer and the aggregate of global potentials.

7.3.2 ADMM Features

The goal of activation is to determine which variables are most likely to change in a future inference. From the analysis in the previous section, we can identify several basic elements for each variable in the model that serve as features for an activation algorithm. For each variable, we have its value at convergence ($\mathbf{y}[i]$), and for each grounding g, the weight (w_g), the value of the potential ($f_g(\mathbf{x}, \tilde{\mathbf{y}}_g)$), and the Lagrange multipliers ($\boldsymbol{\alpha}_g[i]$) measuring the aggregate deviation from consensus. We discuss each of these features to motivate their importance in an activation algorithm.

The value of a variable at convergence can provide a useful signal in certain situations, where a model has clear semantics. For example, the formulation of HL-MRFs often lends itself to a logical interpretation with binary outcomes, as in the cases of collective classification of attributes that are either present or absent. In this setting, assignments in the vicinity of 0.5 represent uncertainty, and therefore provide good candidates for activation. Unfortunately, this feature is not universal. Many successful HL-MRF models adopt semantics that use continuous values to model continuous variables, such as pixel intensity in image completion tasks or Likert-scale ratings in recommender systems. In this case, the semantics of the variable's consensus value may provide an ambiguous signal for activation.

The weighted potentials of each variable contribute directly to the probability of the MAP configuration. Since the log-probability is proportional to the *negated* energy, $-E$, high weights and high potential values decrease the probability of the assignment. Intuitively, activating those variables that contribute high weighted potentials provides

the best mechanism for approaching the full inference MAP state. A complication to this approach is that each weighted potential can depend on many variables. However, the potential value is a scalar quantity and there is no general mechanism to apportion the loss to the contributing variables.

In contrast, the Lagrange multipliers provide a granular perspective on each variable's effect on Equation 7.13. For each variable copy ($\tilde{\mathbf{y}}_g$), the Lagrange multiplier aggregates the difference between the copy and the global consensus across iterations. High Lagrange multipliers signal discord between the local minimizer and the global minimizer, indicating volatility. Activating variables with high Lagrange multipliers can resolve this discord in future inference using updated evidence. However, updated evidence may also resolve the disagreement between the local and global minimum, obviating an update to the variable.

7.3.3 Activation Algorithms

Building on our analysis of ADMM optimization, we introduce two activation algorithms for online collective inference, "agnostic activation" and "relational activation". Both algorithms produce a ranking that prioritizes each variable for inclusion in inference. The key difference between these algorithms is whether new or updated evidence is an input to the algorithm. Agnostic activation scores variables concurrently with inference, based on their susceptibility to change in future inferences. In contrast, relational activation runs prior to inference, with scores based primarily on relationships between variables and updated evidence in the factor graph.

Each approach has different advantages. Agnostic activation scores variables during inference, providing a performance advantage since the scoring algorithm does not delay a future run of inference. However, this technique has a slower response to new evidence since scoring occurs before such evidence is available. Relational activation can respond to newly-arrived evidence and choose variables related to new evidence, but this requires delaying scoring which can add a computational overhead to inference.

Both activation algorithms output a ranking of the variables, which requires a scoring function. We introduce two scoring functions that use the ADMM features described Section 7.3.2. Our first scoring function, VALUE, captures the intuition that uncertain variables are valuable activation candidates using the function $1 - |0.5 - \mathbf{y}[i]|$, where $\mathbf{y}[i]$ is the consensus value for variable i. The second scoring function, WLM, uses both the weight and Lagrange multipliers of each potential. For each variable, we define a set of weighted Lagrange multiplier magnitudes, $\mathcal{A}_w[i] \triangleq \{|w_g \boldsymbol{\alpha}_g[i]|\}$. To obtain a single scalar score, we take the maximum value of $\mathcal{A}_w[i]$.

The agnostic activation algorithm simply ranks each variable by their score from a scoring function, irrespective of the new evidence. The RELATIONAL algorithm combines the score with information about the new evidence. Using the existing ground model, RELATIONAL first identifies all ground potentials dependent on the new evidence. Then, using these ground potentials as a seed set, the algorithm performs a breadth-first search of the factor graph adding the variables involved in each factor it encounters to the frontier. Traversing the factor graph can quickly identify many candidate variables, so we prioritize variables in the frontier by $\frac{S}{2^d}$ where S is the score assigned by a scoring function and d is the minimum distance between the variable and an element of the seed set in the factor

graph.

The ranking output by either agnostic or relational activation lets us prioritize which variables to activate. Given a budget for the number or percentage of variables to infer, we activate a corresponding number of variables from the ranking. The remaining variables are constrained to their previously inferred values. We selectively ground the model, including only those rules that involve an activated variable. Following inference on the ground model, we use the updated optimization state to produce new scores.

When an inactive variable is treated as a constant, it does not have any associated Lagrange multipliers, and lacks features for the WLM scoring function. Therefore, instead of treating fixed variables as constants, we introduce them as constrained variables in the optimization. This allows us to generate features by capturing the discrepancy between a variable's constrained value and the value of its local copies in groundings involving activated variables.

Our implementation of the agnostic activation algorithm is extremely efficient; all necessary features are byproducts of the inference optimization. Once scores are computed and the activated atoms are selected, the optimization state can be discarded to avoid additional resource commitments. In relational activation, scoring is similarly efficient, but there is an additional overhead of preserving the ground model to allow fast traversal of the factor graph. By selectively grounding the model, we replace queries that scan the entire database, potentially many times, with precise queries that exploit indices for faster performance. Finally, selectively activating atoms produces an optimization objective with fewer terms, allowing quicker optimization.

7.4 Evaluation

To better understand the regret bounds and approximation algorithms for online inference, we perform an empirical evaluation on two online collective inference settings. The first setting is a synthetic online collective classification task where the data generator allows us to modulate the importance of collective dependencies and control the amount of noise. The second evaluation setting is a real-world collaborative filtering task, where user preferences are incrementally revealed and the outputs of a recommender system are correspondingly updated. In order to support repeated full inference of all variables, both of these datasets are necessarily small.

In both evaluation settings, we measure regret relative to full inference and inference error relative to ground truth. The results demonstrate that empirical regret follows the form of our regret bounds. We also evaluate the approximation algorithms presented in Section 7.3.3, to investigate whether features from the optimization algorithm can reliably determine which variables to activate. The results show that our approximation algorithms are able to reduce running time by upwards of 65%, with inference regret relative to full inference.

All experiments are implemented using the open-source PSL framework and our code is available on GitHub.[1]

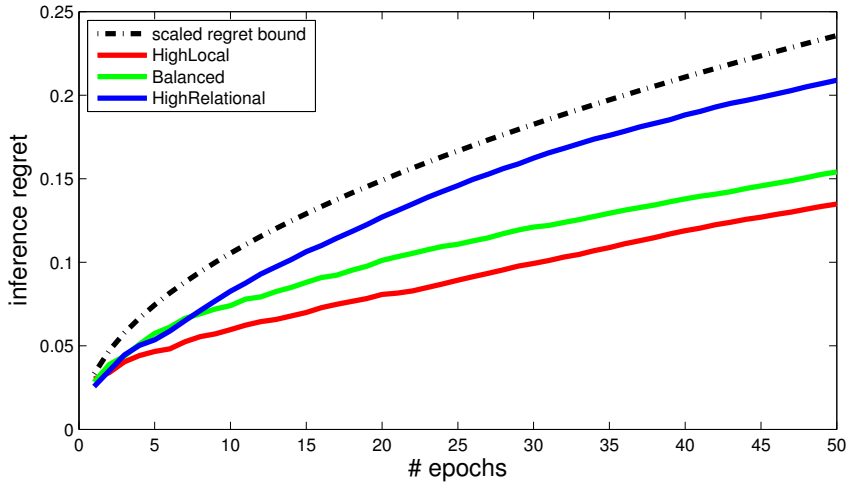
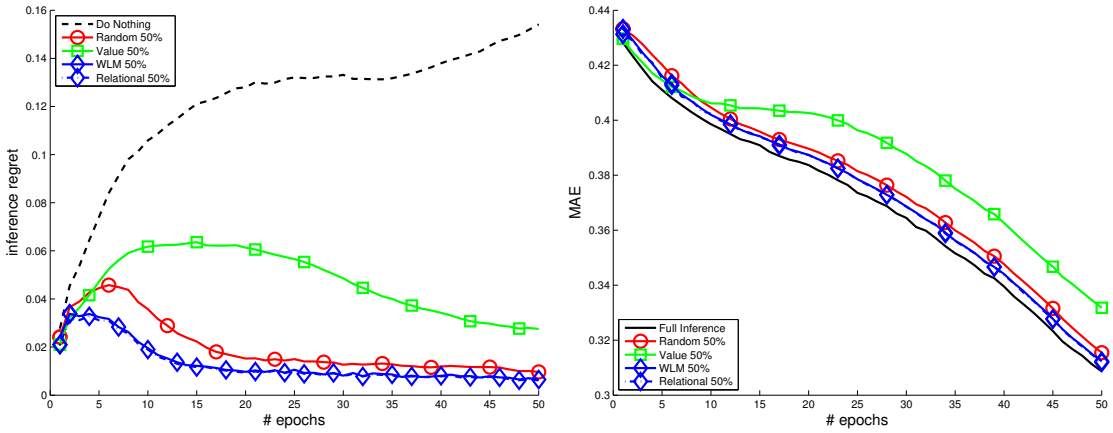


Figure 7.1: Inference regret, w.r.t. full inference, of fixing the original MAP state (i.e., no updates) in the HIGHLOCAL, HIGHCOLLECTIVE and BALANCED data models.



(a) Inference Regret (b) MAE

Figure 7.2: Inference regret (w.r.t. full inference) and MAE (w.r.t. ground truth) using various approximation algorithms, with 50% activation, in the COMPLEX data model.

7.4.1 Online Collective Classification

Our evaluation data simulates a collective classification problem of inferring labels for users in a social network as new evidence is incrementally revealed. Each user is assigned one of two mutually exclusive labels. Some portion of the users have observed labels,

[1] https://github.com/puuj/uai15-boci-code

while the labels of the remaining users are inferred. At each epoch, the label of one more user is revealed, so the model must update the inferred labels for the remaining users with unknown labels.

For each user, we generate local and collective features correlated with the user's label. Local features are generated for each user and label by drawing from a Gaussian distribution conditioned on the label, such that the mean is t for the true label and $1-t$ for the incorrect label. The collective features are links between users, generated randomly using the following process: for each pair of users with the same label, a link is generated with probability p; for each pair of users with different labels, a link is created with probability $1-p$. We refer to p as the affinity of the network.

We model the data using the PSL rules described in Section 7.1 and learn weights for the model. Varying the parameters of the data generator impacts inference in the learned model, since the learned weights are proportional to the discriminative power of their associated rules. For example, varying the distance between the conditional means of the local features controls the importance of the local evidence rule: when the means are far apart, local evidence has high discriminative power; however, when the means are close, local evidence does not provide much signal.

We introduce three data models: HIGHLOCAL ($t = .8, p = .75$), HIGHCOLLECTIVE ($t = .55, p = .9$), and BALANCED ($t = .7, p = .75$). We combine these three conditions in a fourth data model, COMPLEX, which samples uniformly from the three settings on a per-user basis resulting in heterogeneous evidence. For each condition, we generate 10 trials, each with a training social network used to learn the model parameters and a separate test social network to evaluate inference quality. Both the training and test

graph have 100 users, with 60 observed user labels in the training graph and 10 observed user labels in the test graph. To infer user attributes, we use the simple collective classification model introduced in Section 7.1. We simulate the process of online inference by creating a sequence of observations consisting of 50 epochs. In each epoch, the true label of a previously unknown user is revealed, resulting in 60 observed user labels at the end of the sequence. For each trial, we generate 10 such sequences from a breadth-first traversal of the network from a randomly chosen user, resulting in a total of 5000 inferences.

In the first experiment, shown in Figure 7.1 we measure the inference regret of fixing variables to the initial MAP state (i.e., not updating inference) over 50 epochs, comparing the HIGHLOCAL, HIGHCOLLECTIVE and BALANCED conditions. Our theoretical analysis predicts that the worst-case regret grows at rate $\mathrm{O}\left(1/\sqrt{\text{epoch}}\right)$. The experimental results exhibit the same growth rate, which is very pronounced for the HIGHCOLLECTIVE data model, where variables are strongly interdependent, and less so for HIGHLOCAL, where variables are largely independent. The key insight is that the collective nature of the inference task determines the regret of online updates.

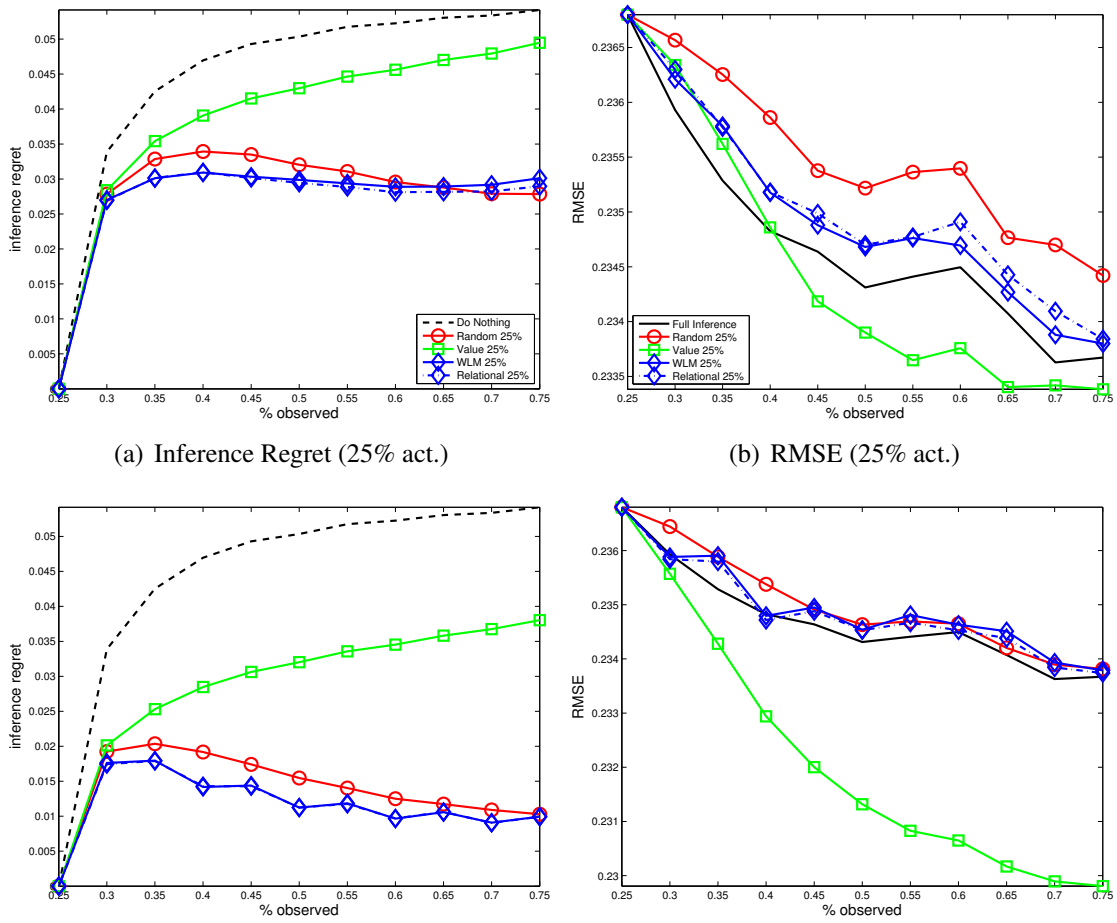
In the second experiment (Figure 7.2), we compare the approximate scoring algorithms with a budget of 50% of unknowns to running full inference on the COMPLEX network. We measure significance across 100 total sequences using a paired t-test with rejection threshold $.05$. For inference regret, we compare against the static algorithm, DONOTHING, which does not update the MAP state, and a random baseline, RANDOM, that fixes an arbitrary subset of 50% of the variables. We compare these to three approximation algorithms described in Section 7.3: VALUE, which uses the value assigned to the variable; WLM, which uses the maximum of the weighted Lagrange multipliers; and

RELATIONAL, which uses WLM to prioritize exploration.

All methods exhibit low regret relative to full inference, contrasting the high regret of the static algorithm, although VALUE exhibits somewhat higher regret. The WLM and RELATIONAL methods have significantly lower regret relative to RANDOM, in 98% and 100% of epochs, respectively. We also compare the mean average error (MAE), with respect to ground truth, of using full inference vs. the approximations. This illustrates that the approximation algorithms remain competitive with full inference, although VALUE again lags in accuracy. Here, the WLM and RELATIONAL methods have significantly lower error than RANDOM in 80% and 100% of epochs, respectively. Comparing the running times highlights the computational benefit of using the approximation algorithms. The average running time for a single trial (which includes training and 10 random sequences of revealed variables) using full inference is 3076 seconds, while approximate inference requires only 955 seconds, a reduction of 69%, with inference time varying less than 3% across methods.

7.4.2 Collaborative Filtering

Our second evaluation task is a collaborative filtering task that employs a collective model to infer the preferences of users. We use the Jester dataset (Goldberg et al., 2001) which includes ratings from 24,983 users on a set of 100 jokes. The task in this setting is to infer the user's rating of each joke. We use the model from Bach et al. (2013) which assigns ratings to jokes based on the joke's similarity to other jokes rated highly by the user. Joke similarity is measured using the mean-adjusted cosine similarity of the observed ratings



(a) Inference Regret (25% act.)

(b) RMSE (25% act.)

(c) Inference Regret (50% act.)

(d) RMSE (50% act.)

Figure 7.3: Inference regret (w.r.t. full inference) and RMSE (w.r.t. ground truth) for the Jester dataset.

of two jokes. (Refer to Bach et al. (2013) for further model details.) We sample 200 users who have rated all 100 jokes and split them into 100 training users and 100 testing users. We generate 10 sequences, each of which consists of a training and testing phase. Model weights are learned using 75% of the training users' ratings observed. During testing, we incrementally reveal $[25\%, 30\%, 40\%, \ldots, 75\%]$ of the testing users' ratings, performing online collective inference at each epoch.

We compare inference regret, relative to full inference, for the RANDOM, VALUE,

WLM and RELATIONAL approximate methods. We also plot the RMSE, relative to ground truth, for full inference and all approximate methods. Figure 7.3a-b show results for 25% activation, and Figure 7.3c-d show 50% activation. Inference regret follows a similar pattern for both budgets, with VALUE showing increasing regret over epochs, and the remaining methods exhibiting level or diminishing regret after the first few epochs. The high regret for VALUE can be explained by considering the RMSE—VALUE actually *improves* the results of full inference, incurring high regret but low RMSE. Our intuition for this improvement is that VALUE fixes polarized user ratings and allows these ratings to have greater influence on other unknown ratings, while full inference produces more moderate ratings for the entire set. The other approximation algorithms remain close to the full inference RMSE (at 50% activation) or perform slightly worse (at 25% activation). Comparing the running times, we find a similar improvement in speed. The average time for a sequence using full inference is 137 seconds, while the approximate methods require only 46 seconds, yielding a speedup of 66%. Approximation methods had consistent timing, varying less than 6%.

7.5 Discussion

In this chapter, we introduce a new problem, *budgeted online collective classification*, which addresses a common problem setting where online inference is necessary but full inference is infeasible, thereby requiring approximate inference updates. Our contributions are: (1) a formal analysis of online collective inference, introducing the concept of inference regret to measure the quality of the approximation; (2) analytic upper bounds

on the inference regret incurred by strongly convex inference; and (3) several algorithms to address the practical problem of activation (i.e., choosing which variables to infer at each epoch), through a close analysis of the MAP inference optimization. The empirical results demonstrate that our activation algorithms exhibit low inference regret and error that is competitive with full inference, while reducing the time required for inference by 65% or more.

This work inspires many exciting areas of future research. One open question is whether one can derive a tighter regret bound using the mechanics of the activation strategy, thus characterizing how performance degrades as a function of the budget. We are also interested in training an "optimal" activation policy that is trained using the variables whose values change the most during full inference. Finally, a crucial assumption in our analysis is that the model structure is fixed, but it is useful to consider the setting in which the set of variables change over time, allowing us to address situations such as new users joining a social network.

Chapter 8: Conclusion and Future Work

In this dissertation, I have developed a framework for knowledge graph construction that addresses many of the practical challenges confronting knowledge base construction systems. Knowledge graph construction requires overcoming many pathological errors in information extraction, a problem I address through the formulation of knowledge graph identification. Implementations of knowledge graph identification must incorporate statistical features from information extractions systems and ontological constraints, and I develop a knowledge graph identification model that is capable of using both these sources of information. Entity resolution is a significant hurdle for knowledge graph construction, and I develop a general model for entity resolution that exploits the relational features in a knowledge graph and is adaptable to many different scenarios.

While the models I develop integrate many powerful features, a key concern is that such models will not scale to realistic problem settings. My choice of hinge-loss Markov random fields alleviates scalability concerns by framing the knowledge graph identification problem as convex optimization. I further improve performance by developing a distributed version of knowledge graph identification.

Finally, I explore what I believe is a new frontier for probabilistic models: streaming (or online) inference. Many real-world problem settings, such as knowledge graph

construction, require updating the inferences of a probabilistic model with new evidence. However, scant research has been devoted to practical approaches for adapting inference results in response to evidence. My work on online collective inference addresses this important area. I introduce inference regret to quantify the consequences of making an approximate update to inference results. By bounding inference regret, I show that such updates are feasible and can preserve inference quality when the updated variables are carefully chosen. I devise several algorithms for updating inference that show attractive empirical performance.

8.1 Future Work

The pursuit of knowledge has tantalized humanity for ages, and remains among the central challenges in artificial intelligence research. Many open questions and practical challenges still confront knowledge graph construction. Two areas where I see great rewards from further research are unifying diverse approaches in the knowledge base construction community, extending my work on online inference to address a broader set of problem settings, and exploring potential applications of the algorithms I devised for online inference to active approaches for probabilistic inference.

While I have been working on knowledge graph identification, a number of different and very promising approaches to information extraction, natural language, understanding and knowledge base construction have been developed. One such development has been the rise of vector space models of language. These models, often trained with recurrent neural networks, capture a latent representation of entities, and use vector operations to

determine the relationships between entities. These features could easily be incorporated into knowledge graph identification, and might provide a unique signal to complement the inputs from an information extraction system. Other popular models in knowledge base construction include matrix factorization approaches that are capable of populating many missing relationships between entities and locally-grounded, random walk based inference approaches. Finding the right way to integrate these approaches with knowledge graph identification while preserving the unique strengths of each remains a compelling open problem.

The second extension to my dissertation research that seems extremely promising is expanding online collective inference to accommodate a more diverse set of problem settings. In my work on online inference, a key limitation is that the structure of the graphical model and the inferred variables are both fixed. This constraint poses problems for many real-world problems which require open-world reasoning, such as when a new user or a new item is added to a recommender system, or a new entity is added to a knowledge graph. I believe that extending online collective inference to include open-world reasoning under certain circumstances may be straightforward. The main obstacle is that the variables in the probability distribution change over time, and I suspect the key to this obstacle may lie in work on default reasoning and lazy inference. By identifying the conditions under which inference can be updated with new variables without requiring a recomputation of the entire MAP state, I anticipate the bound on inference regret can be applied to a broader set of problem settings, and this may in turn yield new insights into algorithms for online inference.

A third area I hope to explore in future work is the potential of active approaches to

probabilistic inference. The problem setting I would like to address is one where evidence can be actively acquired by consulting an oracle, which may take the form of a label provided by a human or a value generated through an expensive computation. The expense of acquiring labels constrains the system to only request the most useful labels. Additionally, once labels are acquired, inference must be repeated using the new evidence. This setting bears many similarities to the online setting. Approximate, partial inference updates may be the key to tractablity in this setting. However, the same methods used for online inference have another application – choosing which labels to acquire. One possibility is using the algorithms designed to activate variables for online inference to choose which labels to actively query. In exploring this setting, it may be the case that the algorithms that are most effective for choosing informative labels to acquire differ from the algorithms that select the best variables to infer. Understanding the differing performance of these algorithms may lend new insight into my existing approach to online inference, as well as inspire new algorithms for approximate online inference.

Appendix A: Sample PSL Program for Knowledge Graph Identification

The following code demonstrates a very simple, but complete PSL program that implements some of the features of knowledge graph identification. The code is written in Groovy, an easily-specified, interpreted scripting language that compiles to Java code. The use of Groovy as an interface to PSL allows users to easily specify and manipulate models using an intuitive syntax rather than dealing with the complex object model defined in the full Java software package. Detailed comments that explain the program follow the code.

```
1  \\Instantiate datastore and model
2  ConfigBundle emptyConfig = new EmptyBundle();
3  DataStore datastore = new RDBMSDataStore(
4                          new H2DatabaseDriver(Type.Disk,
5                          '/tmp/psl', true, emptyConfig);
6  PSLModel model = new PSLModel(this, datastore);
7
8  \\define model predicates
9  ArgumentType uid = ArgumentType.UniqueID;
10 model.add predicate: "Lbl", types: [uid, uid];
11 model.add predicate: "CandLbl", types: [uid, uid];
12 model.add predicate: "Rel", types: [uid, uid, uid];
13 model.add predicate: "CandRel", types: [uid, uid, uid];
14 model.add predicate: "Dom", types: [uid, uid];
15 model.add predicate: "Mut", types: [uid, uid];
16 model.add predicate: "Sub", types: [uid, uid];
17
18 \\ define model rules
19 model.add rule: ~Lbl(E,L), weight : 0.5;
20 model.add rule: ~Rel(E1,E2,R), weight : 0.5;
```

```
21 model.add rule: ( CandLbl(E,L) ) >> Lbl(E,L),
22                    weight : 1;
23 model.add rule: ( CandRel(E1,E2,R) ) >> Rel(E1,E2,R),
24                    weight : 1;
25 model.add rule: ( Dom(R,L) & Rel(E1,E2,R) ) >> Lbl(E1,L),
26                    constraint:1;
27 model.add rule: ( Mut(L1,L2) & Lbl(E,L1) ) >> ~Lbl(E,L2),
28                    constraint:true;
29 model.add rule: ( Sub(L1,L2) & Lbl(E,L1) ) >> Lbl(E,L2),
30                    constraint:true;
31
32 \\load data and set up database
33 Partition inferences = datastore.getPartition("output");
34 Partition evidence = datastore.getPartition("input");
35
36 def pNames = ["CandLbl", "CandRel", "Dom", "Mut", "Sub"];
37 PredicateFactory pFactory = PredicateFactory.getFactory();
38 for( String pName : pNames ){
39   Predicate p = pFactory.getPredicate(pName);
40   Inserter inserter = datastore.getInserter(p,evidence);
41   InserterUtils.loadDelimitedData(inserter, pName+".tsv");
42 }
43
44 Database inferenceDB = data.getDatabase(inferences,
45           [CandLbl, CandRel, Domain, Mut, Sub], evidence);
46
47 \\Run inference
48 MPEInference mpe = new LazyMPEInference(model,
49                           inferenceDB,
50                           emptyConfig);
51 InferenceResult result = mpe.optimize();
52
53 mpe.close(); inferenceDB.close(); datastore.close();
```

Detailed Comments

Line 3 of the program instantiates a datastore, PSL's mechanism for interacting with databases. The database implementation in this case is a relational database management system. The arguments to the constructor specify that an H2 Database should be used, and a new database file created at the path `\tmp\psl`, with no additional configuration

modification.

Line 6 instantiates a PSL model, and in lines 10-16, the logical predicates that define the model are defined. While the arguments to the predicate can take many possible types (such as String, Double, Integer, Date), in this model all arguments are unique identifiers. For readability, this argument type is defined on line 9.

The next component of a PSL model is the specification of logical rules that determine relationships between variables. Line 19 and 20 contain rules specifying negative priors for the `Lbl` and `Rel` predicates, respectively. Note that the symbol specifies negation in PSL. The negative prior enforces that any fact unsupported by evidence will have a false value. To avoid overwhelming evidence, the negative prior is given a low weight (0.5).

Lines 21-24 link the facts in the knowledge graph to candidate extractions in the evidence. In practice, a variety of different techniques would be used to generate these extractions, and each technique would correspond to a separate pair of rules. In this simple example, only one pair of rules is used. These rules are given a low weight of 1.0, although this weight is higher than the prior. In a more sophisticated program, the weight of each rule would be learned.

Lines 25-30 introduce ontological constraints in the model. Lines 25-26 contain a rule expressing the Domain constraint: if the domain of a relation `R` is `L`, and the relation `R` holds between entities `E1` and `E2`, then the entity `E1` has label `L`. Line 26 specifies that this rule is to be treated as a constraint: any feasible solution to the inference problem must obey this constraint. A second way of thinking of constraints is that they are rules with infinite weight; the probability of any interpretation with an unsatisfied constraint

thus approaches 0. Lines 27-28 similarly define the mutual exclusion and subsumption of labels as constraints.

Lines 32-33 define *partitions* of the database for inferred variables and evidence atoms. A partition is a logical division of atoms in the database. Partitions help specify the role of an atom during learning or inference by allowing the user to easily differentiate the atoms that will be used as observed evidence, left unobserved and excluded from the model, or selected as targets for the inference process. Here, I define two partition - one for the output of the model, the inferences of the knowledge graph, and the second as the input to the model of extracted candidates and ontological relationships. These partitions are references through the names "output" and "input" respectively.

In lines 36-42, I load data into the database. Line 36 specifies the names of the predicates for which data is loaded. Starting on line 38, a for loop iterates over each of these predicate names. On line 39, a Predicate object is retrieved for the predicate that corresponds to the given predicate name. For this predicate, an Inserter object is generated by the datastore on line 40. The inserter can be used to insert data for a specified predicate into a specific partition, in this case the evidence partition. This inserter is used to load data on line 41 with the help of a convenience method from a utility class, InserterUtils. The data is loaded from a file specified as the second argument. In this program, the files are assumed to be in the working directory and named based on the predicate for which they contain data. For example, `CandRel.tsv` would contain data, consisting of a series of lines, each specifying an atom with the arguments delimited by tabs, for the `CandRel` predicate.

Line 44-45 specify a Database object for use during inference. This nomenclature

may be somewhat confusing, since it conflicts with that of databases in the datastore. In PSL, a database is a specification of evidence and inference partitions, accompanied by a set of atoms that are fully observed. In this case, the first argument to the getDatabase function is the "write" partition where new inferences will be written. The second argument is the set of predicates that are fully observed. The consequence of specifying these predicates as fully observed is that PSL will make a closed-world assumption for each predicate, assigning any atom that is absent from the inference and evidence partitions a value of 0. The final argument to the function is the evidence partition. For convenience, PSL allows an arbitrary number of partitions to be specified as evidence, however only one is necessary in this simple program.

Lines 48-51 perform inference in the PSL model. Line 48-50 define the inference object. The arguments to the inference object are the model (which specifies the rules the model will use during inference), the inference database (specifying which atoms to use as evidence and the location for new inferences), and a configuration package, which is unnecessary for this example. The inference package in this example is `LazyMPEInference`. Lazy inference is an iterative procedure where inference targets are computed using available evidence. After inference is completed, the grounding process is repeated using the new inferences as well as the initial evidence. The benefit of lazy inference is that the inference targets do not need to be enumerated and specified in advance. The process of enumerating inference targets may be cumbersome for some users. Additionally, in models where outputs are sparse, lazy inference can improve scalability by reducing the memory footprint of the model. The drawback of lazy inference is that inference must be run repeatedly until no new inferences are possible.

Finally, on line 51, the inference optimization is called. By default, PSL will compile the model and evidence into a set of ground rules, convert each rule into an optimization potential, and then use the ADMM algorithm to perform a joint optimization across these potentials to determine the configuration of variables that minimizes the energy function, thus maximizing the probability of the output.

In practice, it is often convenient to export the inference output from the database to an easily manipulable format, such as a text file. However, for the sake of brevity, this is not included in the example. Line 52 simple closes the objects constructed during the program, committing any output and freeing any resources held by the program to the system.

Appendix B: Additional Results for Knowledge Graph Identification

B.1 Baseline Results

Table B.1: Results for the baseline model on the closed-world knowledge graph identification problem for NELL for all facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.783	0.644	1.000
0.100	0.804	0.677	0.989
0.200	0.804	0.677	0.989
0.300	0.804	0.677	0.989
0.400	0.804	0.677	0.989
0.500	0.742	0.695	0.797
0.600	0.695	0.702	0.689
0.700	0.144	0.987	0.078
0.800	0.144	0.987	0.078
0.900	0.144	0.987	0.078
1.000	0.144	0.987	0.078

Table B.2: Results for the baseline model on the closed-world knowledge graph identification problem for NELL for relation facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.834	0.715	1.000
0.100	0.839	0.724	0.998
0.200	0.839	0.724	0.998
0.300	0.839	0.724	0.998
0.400	0.839	0.724	0.998
0.500	0.735	0.788	0.688
0.600	0.641	0.847	0.515
0.700	0.102	0.990	0.054
0.800	0.102	0.990	0.054
0.900	0.102	0.990	0.054
1.000	0.102	0.990	0.054

Table B.3: Results for the baseline model on the closed-world knowledge graph identification problem for NELL for label facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.713	0.553	1.000
0.100	0.751	0.611	0.975
0.200	0.751	0.611	0.975
0.300	0.751	0.611	0.975
0.400	0.751	0.611	0.975
0.500	0.751	0.611	0.975
0.600	0.751	0.611	0.975
0.700	0.208	0.985	0.116
0.800	0.208	0.985	0.116
0.900	0.208	0.985	0.116
1.000	0.208	0.985	0.116

Table B.4: Comparison of the baseline model and PSL-KGI on the closed-world knowledge graph identification problem for NELL for all facts. The results show a sample of facts with the maximally-differing truth values between the two methods.

Fact	Baseline	PSL-KGI
REL(cubs,playoffs,teamWonTrophy)	0.67	0.00
REL(packers,playoffs,teamWonTrophy)	0.67	0.00
REL(chargers,playoffs,teamWonTrophy)	0.67	0.00
REL(colts,playoffs,teamWonTrophy)	0.67	0.00
REL(aol,bebo,acquired)	0.67	0.00
REL(lakers,playoffs,teamWonTrophy)	0.67	0.00
REL(adobe,adobe_acrobat, producesProduct)	0.67	0.00

B.2 Results Excluding Extractor Source Information

Table B.5: Results for the NoSrcs model on the closed-world knowledge graph identification problem for NELL for all facts. The model does not use different predicates for the different NELL extractors. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.783	0.644	1.000
0.100	0.810	0.687	0.986
0.200	0.810	0.687	0.985
0.300	0.810	0.687	0.985
0.400	0.810	0.687	0.985
0.500	0.810	0.687	0.985
0.600	0.823	0.737	0.932
0.700	0.801	0.848	0.759
0.800	0.536	0.975	0.370
0.900	0.450	0.993	0.291
1.000	0.377	0.994	0.232

Table B.6: Results for the NoSrcs model on the closed-world knowledge graph identification problem for NELL for relation facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.834	0.715	1.000
0.100	0.849	0.742	0.993
0.200	0.849	0.742	0.992
0.300	0.849	0.742	0.992
0.400	0.849	0.742	0.992
0.500	0.849	0.742	0.992
0.600	0.850	0.744	0.992
0.700	0.806	0.792	0.821
0.800	0.349	0.940	0.214
0.900	0.162	0.982	0.088
1.000	0.114	0.982	0.060

Table B.7: Results for the NoSrcs model on the closed-world knowledge graph identification problem for NELL for label facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.713	0.553	1.000
0.100	0.751	0.612	0.974
0.200	0.751	0.612	0.974
0.300	0.751	0.612	0.974
0.400	0.752	0.612	0.974
0.500	0.752	0.612	0.974
0.600	0.775	0.724	0.834
0.700	0.791	0.992	0.658
0.800	0.768	0.996	0.625
0.900	0.768	0.996	0.625
1.000	0.679	0.997	0.514

Table B.8: Comparison of the NoSrcs model and PSL-KGI on the closed-world knowledge graph identification problem for NELL for all facts. The results show a sample of facts with the maximally-differing truth values between the two methods.

Fact	NoSrcs	PSL-KGI
LBL(hash_brown_potatoes,food)	0.54	0.95
LBL(doctor_zhivago,creativework)	0.57	0.97
LBL(doctor_zhivago,movie)	0.57	0.97
LBL(twilight,creativework)	0.61	0.97
LBL(twilight,movie)	0.61	0.97
LBL(sideways,creativework)	0.61	0.97
LBL(sideways,movie)	0.61	0.97
LBL(boogie_nights,creativework)	0.64	0.97

B.3 Results Excluding Entity Resolution Information

Table B.9: Results for the NoER model on the closed-world knowledge graph identification problem for NELL for all facts. The model does not use entity resolution rules or information in the model. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.783	0.644	1.000
0.100	0.810	0.687	0.985
0.200	0.810	0.687	0.985
0.300	0.811	0.690	0.984
0.400	0.827	0.724	0.964
0.500	0.851	0.768	0.955
0.600	0.848	0.787	0.920
0.700	0.821	0.820	0.821
0.800	0.697	0.882	0.576
0.900	0.543	0.896	0.389
1.000	0.414	0.996	0.261

Table B.10: Results for the NoER model on the closed-world knowledge graph identification problem for NELL for relation facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.834	0.715	1.000
0.100	0.849	0.742	0.992
0.200	0.849	0.742	0.992
0.300	0.849	0.742	0.992
0.400	0.849	0.742	0.992
0.500	0.848	0.742	0.989
0.600	0.843	0.757	0.952
0.700	0.807	0.785	0.830
0.800	0.619	0.845	0.488
0.900	0.380	0.779	0.251
1.000	0.090	0.989	0.047

Table B.11: Results for the NoER model on the closed-world knowledge graph identification problem for NELL for label facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.713	0.553	1.000
0.100	0.751	0.612	0.974
0.200	0.751	0.612	0.974
0.300	0.755	0.618	0.969
0.400	0.790	0.694	0.917
0.500	0.857	0.819	0.899
0.600	0.857	0.846	0.867
0.700	0.845	0.889	0.806
0.800	0.811	0.928	0.720
0.900	0.761	0.996	0.616
1.000	0.759	0.997	0.613

Table B.12: Comparison of the NoER model and PSL-KGI on the closed-world knowledge graph identification problem for NELL for all facts. The results show a sample of facts with the maximally-differing truth values between the two methods.

Fact	NoER	PSL-KGI
LBL(ussr,organization)	0.24	1.00
LBL(ampalaya,food)	0.35	0.99
LBL(ampalaya,vegetable)	0.35	0.99
LBL(acc_conference,organization)	0.37	1.00
LBL(acc_conference,sportsleague)	0.37	1.00
LBL(bell_centre,building)	0.40	1.00
LBL(bell_centre,location)	0.42	1.00
LBL(bell_centre,attraction)	0.43	1.00

B.4 Results Excluding Ontological Information

Table B.13: Results for the NoOnto model on the closed-world knowledge graph identification problem for NELL for all facts. The model does not use ontological rules or information in the model. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.783	0.644	1.000
0.100	0.804	0.677	0.989
0.200	0.804	0.677	0.989
0.300	0.804	0.677	0.989
0.400	0.800	0.682	0.969
0.500	0.830	0.815	0.845
0.600	0.816	0.832	0.800
0.700	0.757	0.853	0.681
0.800	0.689	0.880	0.566
0.900	0.217	0.983	0.122
1.000	0.165	0.985	0.090

Table B.14: Results for the NoOnto model on the closed-world knowledge graph identification problem for NELL for relation facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.834	0.715	1.000
0.100	0.839	0.724	0.998
0.200	0.839	0.724	0.998
0.300	0.839	0.724	0.998
0.400	0.834	0.734	0.966
0.500	0.817	0.824	0.809
0.600	0.794	0.852	0.744
0.700	0.699	0.864	0.587
0.800	0.649	0.860	0.521
0.900	0.169	0.982	0.092
1.000	0.086	0.988	0.045

Table B.15: Results for the NoOnto model on the closed-world knowledge graph identification problem for NELL for label facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.783	0.644	1.000
0.100	0.810	0.687	0.985
0.200	0.810	0.687	0.985
0.300	0.811	0.690	0.984
0.400	0.827	0.724	0.965
0.500	0.851	0.767	0.956
0.600	0.848	0.786	0.921
0.700	0.824	0.821	0.827
0.800	0.710	0.884	0.594
0.900	0.570	0.899	0.418
1.000	0.439	0.993	0.281

Table B.16: Comparison of the NoOnto model and PSL-KGI on the closed-world knowledge graph identification problem for NELL for all facts. The results show a sample of facts with the maximally-differing truth values between the two methods.

Fact	NoOnto	PSL-KGI
REL(community_college,baseball, teamPlaysSport)	0.81	0.00
REL(convention_center,anaheim, stadiumInCity)	0.81	0.00
LBL(ncaa,football)	0.80	0.01
LBL(comiskey_park,location)	0.24	1.00
LBL(cardiff_intl_arena,location)	0.24	1.00
LBL(buck_shaw_stadium,building)	0.24	1.00
LBL(buck_shaw_stadium,attraction)	0.24	1.00
LBL(brian_rogers,person)	0.24	1.00
LBL(david_dejesus,person)	0.24	1.00
LBL(moises_alou,person)	0.24	1.00

B.5 Results for the Knowledge Graph Identification Model

Table B.17: Results for the KGI model on the closed-world knowledge graph identification problem for NELL for all facts. The model uses ontological rules, entity resolution rules, and extractor confidence rules. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.783	0.644	1.000
0.100	0.810	0.687	0.985
0.200	0.810	0.687	0.985
0.300	0.811	0.690	0.984
0.400	0.827	0.724	0.965
0.500	0.851	0.767	0.956
0.600	0.848	0.786	0.921
0.700	0.824	0.821	0.827
0.800	0.710	0.884	0.594
0.900	0.570	0.899	0.418
1.000	0.439	0.993	0.281

Table B.18: Results for the KGI model on the closed-world knowledge graph identification problem for NELL for relation facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.834	0.715	1.000
0.100	0.849	0.742	0.992
0.200	0.849	0.742	0.992
0.300	0.849	0.742	0.992
0.400	0.849	0.742	0.992
0.500	0.848	0.742	0.988
0.600	0.843	0.756	0.952
0.700	0.809	0.786	0.834
0.800	0.641	0.849	0.514
0.900	0.425	0.798	0.290
1.000	0.144	0.979	0.077

Table B.19: Results for the KGI model on the closed-world knowledge graph identification problem for NELL for label facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.713	0.553	1.000
0.100	0.751	0.612	0.974
0.200	0.751	0.612	0.974
0.300	0.755	0.618	0.969
0.400	0.792	0.695	0.920
0.500	0.858	0.818	0.903
0.600	0.857	0.845	0.870
0.700	0.849	0.886	0.814
0.800	0.814	0.927	0.725
0.900	0.770	0.994	0.628
1.000	0.761	0.996	0.616

B.6 Results for the Open-World Knowledge Graph Identification Model

Table B.20: Results for the KGI model on the open-world knowledge graph identification problem for NELL for all facts. The model uses ontological rules, entity resolution rules, and extractor confidence rules and does not restrict inferences to the test set. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.783	0.644	1.000
0.100	0.816	0.707	0.964
0.200	0.854	0.773	0.955
0.300	0.851	0.778	0.939
0.400	0.855	0.801	0.916
0.500	0.848	0.826	0.871
0.600	0.818	0.867	0.775
0.700	0.784	0.887	0.703
0.800	0.750	0.915	0.636
0.900	0.721	0.925	0.591
1.000	0.526	0.928	0.367

Table B.21: Results for the KGI model on the open-world knowledge graph identification problem for NELL for relation facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.834	0.715	1.000
0.100	0.846	0.756	0.961
0.200	0.846	0.759	0.957
0.300	0.845	0.761	0.949
0.400	0.844	0.773	0.929
0.500	0.834	0.793	0.880
0.600	0.808	0.837	0.781
0.700	0.766	0.849	0.698
0.800	0.737	0.875	0.637
0.900	0.696	0.883	0.574
1.000	0.411	0.857	0.270

Table B.22: Results for the KGI model on the open-world knowledge graph identification problem for NELL for label facts. The results show the performance soft-truth thresholds.

threshold	F1	Precision	Recall
0.000	0.713	0.553	1.000
0.100	0.770	0.639	0.968
0.200	0.868	0.797	0.952
0.300	0.862	0.810	0.922
0.400	0.874	0.854	0.894
0.500	0.871	0.887	0.856
0.600	0.836	0.923	0.764
0.700	0.815	0.955	0.710
0.800	0.773	0.992	0.634
0.900	0.764	0.997	0.619
1.000	0.689	0.997	0.526

Table B.23: Comparison of the open-world model to the closed-world model for all facts inferred by the open-world. The results show a sample of facts where the open-world truth value is much higher than the closed-world truth value.

Fact	KGI-Complete	KGI
REL(bruins,banknorth_garden, teamhomestadium)	1.00	0.00
REL(bruins,fleet_center,atlocation)	1.00	0.00
REL(bruins,nhl,subpartof)	1.00	0.00
REL(rangers,hockey,teamplayssport)	1.00	0.00
REL(lemur,jamie_callan, mlsoftwareauthor)	1.00	0.00
LBL(lanni,agent)	1.00	0.00
LBL(eibe_frank,person)	1.00	0.00
REL(eibe_frank,weka,involvedwith)	1.00	0.00
REL(washington_d_c,anacostia_museum, citymuseums)	1.00	0.00
LBL(anacostia_museum,building)	1.00	0.00

Table B.24: Comparison of the open-world model to the closed-world model for all facts inferred by the open-world. The results show a sample of facts where the open-world truth value is much lower than the closed-world truth value.

Fact	KGI-Complete	KGI
REL(rfk_memorial_stadium,washington, stadiumincity)	0.00	1.00
REL(charlotte_bobcats, time_warner_cable_arena, teamhomestadium)	0.01	0.96
REL(boston_celtics,los_angeles, teamplaysincity)	0.00	0.94
REL(denver_nuggets,los_angeles, teamplaysincity)	0.00	0.94
REL(cleveland_cavaliers,los_angeles, teamplaysincity)	0.00	0.94
REL(detroit_pistons,los_angeles, teamplaysincity)	0.00	0.94
REL(georgia_tech,basketball, teamplayssport)	0.01	0.95
REL(pittsburgh_steelers,baltimore, teamplaysincity)	0.00	0.94
REL(san_diego_chargers,super_bowl, teamwontrophy)	0.01	0.92
REL(seahawks,super_bowl, teamwontrophy)	0.01	0.92
LBL(congo,country)	0.01	0.86
REL(qwest_field,seattle, stadiumincity)	0.00	0.85
REL(lusaka,zambia,cityincountry)	0.03	0.87

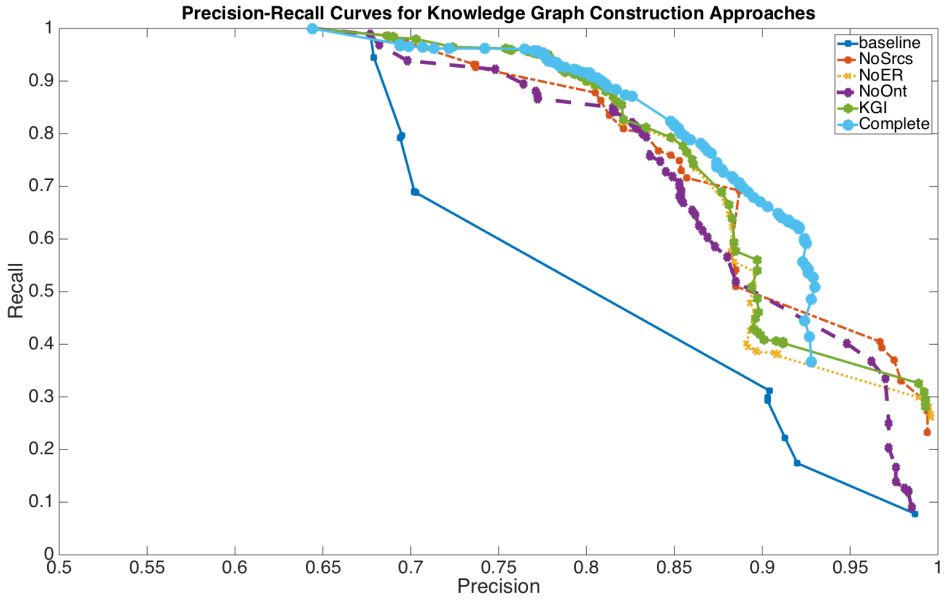


Figure B.1: This figure shows the precision-recall curve for the different knowledge graph construction models. The baseline model, which does not use any collective reasoning, severely underperforms all other approaches. Models that omit the confidence information of uncertain sources (orange squares), entity resolution (yellow exs), or ontological information (purple diamonds) do not perform as well as knowledge graph identification (green hexagons). Complete knowledge graph identification (blue circles) in the open-world setting, infers a complete knowledge graph and demonstrates good performance but suffers from low recall at the highest precision.

Bibliography

U. Acar, A. Ihler, R. Mettu, and O. Sümer. Adaptive Inference on General Graphical Models. In *UAI*, 2008.

U. Acar, A. Ihler, R. Mettu, and O. Sümer. Adaptive Updates for MAP Configurations with Applications to Bioinformatics. *IEEE Statistical Signal Processing (SSP)*, pages 413–416, 2009.

G. Angeli, J. Tibshirani, J. Y. Wu, and C. D. Manning. Combining Distant and Partial Supervision for Relation Extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2014.

G. Antoniou and F. Van Harmelen. *A Semantic Web Primer*. MIT press, 2004.

S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A Nucleus for a Web of Open Data. *The Semantic Web*, pages 722–735, 2007.

F. Baader, I. Horrocks, and U. Sattler. Description Logics as Ontology Languages for the Semantic Web. *Mechanizing Mathematical Reasoning*, pages 228–248, 2005.

S. H. Bach, M. Broecheler, L. Getoor, and D. P. O'Leary. Scaling MPE Inference for Constrained Continuous Markov Random Fields with Consensus Optimization. In *NIPS*, 2012.

S. H. Bach, B. Huang, B. London, and L. Getoor. Hinge-Loss Markov Random Fields: Convex Inference for Structured Prediction. In *UAI*, 2013.

S. H. Bach, M. Broecheler, B. Huang, and L. Getoor. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic. *arXiv preprint*, arXiv:1505.04406 [cs.LG], 2015.

N. Balcan, A. Blum, and Y. Mansour. Exploiting Ontology Structures and Unlabeled Data for Learning. In *ICML*, 2013.

A. Barr and J. Davidson. *Knowledge Representation*, volume 1, pages 141–222. William Kaufmann, Los Altos, CA, 1981.

T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific american*, 284 (5):28–37, 2001.

I. Bhattacharya and L. Getoor. Collective Entity Resolution in Relational Data. *ACM Transactions on Knowledge Discovery and Datamining*, 1(1), 2007.

M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48. ACM, 2003.

C. Bizer and A. Seaborne. D2RQ–Treating Non-RDF Databases as Virtual RDF Graphs. In *ISWC*, 2004.

C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. {DBpedia} - a Crystallization Point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154 – 165, 2009. ISSN 1570-8268. The Web of Data.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

M. Boden. *Mind As Machine: A History of Cognitive Science*. Oxford University Press, Inc., New York, NY, USA, 2008. ISBN 019954316X, 9780199543168.

P. L. Boeuf. FRBR and Further. *Cataloging & classification quarterly*, 32(4):15–52, 2001.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning Via the Alternating Direction Method of Multipliers. *Foundations and Trends Machine Learning*, 3(1):1–122, 2011.

R. d. S. Braz, E. Amir, and D. Roth. Lifted First-Order Probabilistic Inference. In *IJCAI*, 2005.

D. Brickley and L. Miller. FOAF Vocabulary Specification 0.98, 2010. see `http://xmlns.com/foaf/spec/20100809.html`.

M. Broecheler, L. Mihalkova, and L. Getoor. Probabilistic Similarity Logic. In *UAI*, 2010.

B. Buchanan and G. Sutherland. Heuristic Dendral: A Program for Generating Explanatory Hypotheses in Organic Chemistry. Technical report, Dept Of Computer Science, Stanford University, California, 1968.

B. G. Buchanan and E. A. Feigenbaum. Dendral and Meta-Dendral: Their Applications Dimension. *Artificial intelligence*, 11(1):5–24, 1978.

B. G. Buchanan and E. H. Shortliffe. *Rule-Based Expert Systems*. Addison-Wesley, 1984.

W. Buntine. Theory Refinement on Bayesian Networks. In *UAI*, 1991.

A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an Architecture for Never-Ending Language Learning. In *AAAI*, 2010a.

A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr, and T. M. Mitchell. Coupled Semi-Supervised Learning for Information Extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM, 2010b.

H. Chan and A. Darwiche. Sensitivity Analysis in Markov Networks. In *IJCAI*, 2005.

H. Chan and A. Darwiche. On the Robustness of Most Probable Explanations. In *UAI*, 2006.

A. Chechetka and C. Guestrin. Focused Belief Propagation for Query-Specific Inference. In *AISTATS*, pages 89–96, 2010.

J. Christensen, S. Soderland, and O. Etzioni. An Analysis of Open Information Extraction Based on Semantic Role Labeling. In *Proceedings of the sixth international conference on Knowledge capture*, pages 113–120. ACM, 2011.

W. Cohen, P. Ravikumar, and S. Fienberg. A Comparison of String Matching Tasks for Names and Addresses. In *IJCAI Workshop on Information Integration on the Web*, 2003.

M. Collins and T. Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, 2005.

M. Collins and Y. Singer. Unsupervised Models for Named Entity Classification. In *EMNLP*, 1999.

H. Daumé III, J. Langford, and D. Marcu. Search-Based Structure Prediction. *Machine Learning Journal*, 2009.

I. Davis, R. Newman, and B. D'Arcus. Expression of Core FRBR Concepts in RDF, 2005. see `http://vocab.org/frbr/core.html`.

M.-C. De Marneffe, B. MacCartney, C. D. Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, 2006.

R. de Salvo Braz, S. Natarajan, H. Bui, J. Shavlik, and S. Russell. Anytime Lifted Belief Propagation. In *SRL*, volume 9, 2009.

S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The Semantic Web: The Roles of XML and RDF. *Internet Computing*, 4(5):63–73, 2000.

S. Dixon and K. Jacobson. LinkedBrainz - A project to provide MusicBrainz NGS as Linked Data. see `http://linkedbrainz.c4dmpresents.org/`.

X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610. ACM, 2014a.

X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang. From Data Fusion to Knowledge Fusion. *Proceedings of the VLDB Endowment*, 7(10):881–892, 2014b.

R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

C. Elkan and A. Monge. The field matching problem: Algorithms and applications. In *Proc. of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press*, 1996.

O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open Information Extraction from the Web. *Communications of the ACM*, 51(12):68–74, 2008.

O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open Information Extraction: the Second Generation. In *IJCAI*, 2011.

A. Fader, S. Soderland, and O. Etzioni. Identifying Relations for Open Information Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics, 2011.

M. Fan, D. Zhao, Q. Zhou, Z. Liu, T. F. Zheng, and E. Y. Chang. Distant Supervision for Relation Extraction with Matrix Completion. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 839–849, 2014.

E. A. Feigenbaum, B. G. Buchanan, and J. Lederberg. On Generality and Problem Solving: A Case Study Using the Dendral Program. Technical report, Department of Computer Science, Stanford University, California, 1970.

I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

S. Fine, Y. Singer, and N. Tishby. The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning*, 32(1):41–62, 1998.

L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

N. Friedman and M. Goldszmidt. Sequential Update of Bayesian Network Structure. In *UAI*, 1997.

P. Gamallo, M. Garcia, and S. Fernández-Lanza. Dependency-Based Open Information Extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 10–18. Association for Computational Linguistics, 2012.

P. Gardenfors, editor. *Belief Revision*. Cambridge University Press, New York, NY, USA, 1992.

A. Globerson and T. Jaakkola. Fixing Max-Product: Convergent Message Passing Algorithms for MAP LP-Relaxations. In *NIPS*, 2007.

J. Golbeck and M. Rothstein. Linking Social Networks on the Web with FOAF: A Semantic Web Case Study. In *AAAI*, volume 8, pages 1138–1143, 2008.

K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval*, 4(2):133–151, 2001.

P. J. Hayes. In Defense of Logic. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'77, pages 559–565, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc. URL `http://dl.acm.org/citation.cfm?id=1624435.1624559`.

P. Hitzler, M. Krotzsch, and S. Rudolph. *Knowledge Representation for the Semantic Web*. KI, 2009.

I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Expressive Description Logics. In *Logic for Programming and Automated Reasoning*, pages 161–180. Springer Berlin Heidelberg, 1999.

I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From {SHIQ} and {RDF} to Owl: The Making of a Web Ontology Language. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1):7 – 26, 2003.

M. A. Jaro. Probabilistic linkage of large public health data files. *Statistics in medicine*, 14(5-7):491–498, 1995.

S. Jiang, D. Lowd, and D. Dou. Learning to Refine an Automatically Extracted Knowledge Base Using Markov Logic. In *ICDM*, 2012.

D. R. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640, 1996.

G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1), 1998.

G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum. The Yago-Naga Approach to Knowledge Discovery. *ACM SIGMOD Record*, 37(4):41–47, 2009.

A. Kimmig, S. H. Bach, M. Broecheler, B. Huang, and L. Getoor. A Short Introduction to Probabilistic Soft Logic. In *NIPS Workshop on Probabilistic Programming*, 2012.

D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

G. Kobilarov, T. Scott, Y. Raimond, S. Oliver, C. Sizemore, M. Smethurst, C. Bizer, and R. Lee. Media Meets Semantic Web–How The BBC uses DBpedia and Linked Data to Make Connections. In *ESWC*, 2009.

D. Koller and N. Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.

S. Krause, H. Li, H. Uszkoreit, and F. Xu. Large-Scale Learning of Relation-Extraction Rules with Distant Supervision from the Web. *International Semantic Web Conference*, pages 263–278, 2012.

M. Krötzsch, F. Simancik, and I. Horrocks. Description Logics. *Intelligent Systems*, 29 (1):12–19, 2014.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, 2001.

K. Laskey. Sensitivity Analysis for Probability Assessments in Bayesian Networks. In *UAI*, 1993.

D. Lenat and R. V. Guha. Cyc: A Midterm Report. *AI magazine*, 11(3):32, 1990.

D. B. Lenat. Cyc: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

D. B. Lenat, M. Prakash, and M. Shepherd. Cyc: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks. *AI magazine*, 6(4):65, 1985.

W. Li, P. van Beek, and P. Poupart. Performing Incremental Bayesian Inference by Dynamic Model Counting. In *AAAI*, 2006.

B. London, B. Huang, B. Taskar, and L. Getoor. Collective Stability in Structured Prediction: Generalization from One Example. In *ICML*, 2013.

B. London, B. Huang, B. Taskar, and L. Getoor. PAC-Bayesian Collective Stability. In *AIStats*, 2014.

L. Màrquez, X. Carreras, K. C. Litkowski, and S. Stevenson. Semantic role labeling: an introduction to the special issue. *Computational linguistics*, 34(2):145–159, 2008.

Mausam, M. D. Schmitz, R. E. Bart, S. Soderland, and O. Etzioni. Open Language Learning for Information Extraction. In *EMNLP*, 2012.

K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

D. Nadeau and S. Sekine. A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

N. Nakashole, M. Theobald, and G. Weikum. Scalable Knowledge Harvesting with High Precision and High Recall. In *Proceedings of the fourth ACM International Conference on Web Search and Data Mining*, pages 227–236. ACM, 2011.

G. M. Namata, S. Kok, and L. Getoor. Collective Graph Identification. In *KDD*, 2011.

A. Nath and P. Domingos. Efficient Belief Propagation for Utility Maximization and Repeated Inference. In *AAAI*, 2010.

G. Navarro. A Guided Tour to Approximate String Matching. *ACM Comput. Surv.*, 33 (1):31–88, Mar. 2001. ISSN 0360-0300. doi: 10.1145/375360.375365. URL `http://doi.acm.org/10.1145/375360.375365`.

A. Newell, J. C. Shaw, and H. A. Simon. Report on a General Problem-Solving Program. In *IFIP Congress*, pages 256–264, 1959.

T.-V. T. Nguyen and A. Moschitti. End-to-End Relation Extraction Using Distant Supervision from External Semantic Repositories. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 277–282. Association for Computational Linguistics, 2011.

M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816, 2011.

M. Nickel, X. Jiang, and V. Tresp. Reducing the Rank in Relational Factorization Models by Including Observable Patterns. *Advances in Neural Information Processing Systems*, pages 1179–1187, 2014.

M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A Review of Relational Machine Learning for Knowledge Graphs: From Multi-Relational Link Prediction to Automated Knowledge Graph Construction. *arXiv preprint arXiv:1503.00759*, 2015.

N. J. Nilsson. *Principles of Artificial Intelligence*. Springer Science & Business Media, 1982.

N. J. Nilsson. Logic and Artificial Intelligence. *Artificial Intelligence*, 47(1):31–56, 1991.

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*, 8(3):42–73, 2012a.

F. Niu, C. Zhang, C. Ré, and J. W. Shavlik. DeepDive: Web-Scale Knowledge-Base Construction Using Statistical Learning and Inference. *VLDS*, 12:25–28, 2012b.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Reasoning*. Morgan Kaufmann Publishers, 1988.

A. Platanios, A. Blum, and T. M. Mitchell. Estimating accuracy from unlabeled data. In *In Proceedings of UAI*, 2014.

J. Pujara, H. Miao, and L. Getoor. Joint Judgments with a Budget: Strategies for Reducing the Cost of Inference. In *ICML Workshop on Machine Learning with Test-Time Budgets*, 2013a.

J. Pujara, H. Miao, L. Getoor, and W. Cohen. Ontology-Aware Partitioning for Knowledge Graph Identification. In *CIKM Workshop on Automatic Knowledge Base Construction*, 2013b.

J. Pujara, H. Miao, L. Getoor, and W. Cohen. Knowledge Graph Identification. In *ISWC*, 2013c.

J. Pujara, K. Murphy, X. L. Dong, and C. Janssen. Probabilistic Models for Collective Entity Resolution Between Knowledge Graphs. In *Bay Area Machine Learning Symposium*, 2014.

J. Pujara, B. London, and L. Getoor. Budgeted Online Collective Inference. In *Uncertainty in Artificial Intelligence*, 2015a.

J. Pujara, H. Miao, L. Getoor, and W. Cohen. Using Semantics & Statistics to Turn Data into Knowledge. *AI Magazine*, 36(1):65–74, 2015b.

Y. Raimond, S. Abdallah, and M. Sandler. The Music Ontology. In *International Conference on Music Information Retrieval*, 2007.

A. L. Rector. Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In *Proceedings of the 2nd International Conference on Knowledge Capture*, pages 121–128. ACM, 2003.

M. Richardson and P. Domingos. Markov Logic Networks. *Machine Learning*, 62(1-2), 2006.

B. Roth, T. Barth, M. Wiegand, and D. Klakow. A Survey of Noise Reduction Methods for Distant Supervision. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, pages 73–78. ACM, 2013.

S. Rudolph. Foundations of Description Logics. In *Reasoning Web: Semantic Technologies for the Web of Data*, pages 76–136. Springer Berlin Heidelberg, 2011.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995.

S. Sarawagi. Information Extraction. *Foundations and Trends in Databases*, 1(3):261–377, Mar. 2008. ISSN 1931-7883. doi: 10.1561/1900000003. URL `http://dx.doi.org/10.1561/1900000003`.

N. Shadbolt, W. Hall, and T. Berners-Lee. The Semantic Web Revisited. *Intelligent Systems*, 21(3):96–101, 2006.

E. H. Shortliffe. A Rule-Based Computer Program for Advising Physicians regarding Antimicrobial Therapy Selection. In *Proceedings of the 1974 annual ACM conference*, volume 2, pages 739–739. ACM, 1974.

E. H. Shortliffe. *Mycin: Computer-Based Medical Consultations*. Elsevier, New York, 1976.

B. Smith, W. Ceusters, B. Klagges, J. Köhler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A. L. Rector, and C. Rosse. Relations in biomedical ontologies. *Genome biology*, 6(5):R46, 2005.

R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with Neural Tensor Networks for Knowledge Base Completion. In *Advances in Neural Information Processing Systems*, 2013.

F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Large Ontology from Wikipedia and Wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6 (3):203–217, 2008.

O. Sümer, U. Acar, A. Ihler, and R. Mettu. Adaptive Exact Inference in Graphical Models. *JMLR*, 12:3147–3186, 2011.

M. Surdeanu, D. McClosky, J. Tibshirani, J. Bauer, A. X. Chang, V. I. Spitkovsky, and C. D. Manning. A Simple Distant Supervision Approach for the Tac-Kbp Slot Filling Task. In *Proceedings of the Text Analysis Conference Workshop*, 2010.

S. Takamatsu, I. Sato, and H. Nakagawa. Reducing Wrong Labels in Distant Supervision for Relation Extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 721–729. Association for Computational Linguistics, 2012.

N. Tandon, G. de Melo, F. Suchanek, and G. Weikum. Webchild: Harvesting and Organizing Commonsense Knowledge from the Web. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pages 523–532. ACM, 2014a.

N. Tandon, G. de Melo, and G. Weikum. Acquiring Comparative Commonsense Knowledge from the Web. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 166–172. AAAI Press, 2014b.

B. Taskar, C. Guestrin, and D. Koller. Max-Margin Markov Networks. In *NIPS*, 2003.

P. Thomas, I. Solt, R. Klinger, and U. Leser. Learning Protein-Protein Interaction Extraction Using Distant Supervision. In *Workshop on Robust Unsupervised and Semi-Supervised Methods in Natural Language Processing*, pages 34–41, 2011.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1995.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *ICML*, 2004.

F. van Harmelen, F. van Harmelen, V. Lifschitz, and B. Porter. *Handbook of Knowledge Representation*. Elsevier Science, San Diego, USA, 2007. ISBN 0444522115, 9780444522115.

C. Wagner. Breaking the Knowledge Acquisition Bottleneck Through Conversational Knowledge Management. *Information Resources Management Journal*, 19(1):70–83, Jan. 2006. ISSN 1040-1628. doi: 10.4018/irmj.2006010104. URL `http://dx.doi.org/10.4018/irmj.2006010104`.

R. A. Wagner and M. J. Fischer. The String-to-String Correction Problem. *J. ACM*, 21(1):168–173, Jan. 1974. ISSN 0004-5411. doi: 10.1145/321796.321811. URL `http://doi.acm.org/10.1145/321796.321811`.

M. Wainwright. Estimating the "Wrong" Graphical Model: Benefits in the Computation-Limited Setting. *JMLR*, 7:1829–1859, 2006.

W. Wang, R. Besançon, O. Ferret, and B. Grau. Filtering and Clustering Relations for Unsupervised Information Extraction in Open Domain. In *Proceedings of the 20th ACM international conference on Information and Knowledge Management*, pages 1405–1414. ACM, 2011.

W. Y. Wang, K. Mazaitis, and W. W. Cohen. ProPPR: Efficient First-Order Probabilistic Logic Programming for Structure Discovery, Parameter Learning, and Scalable Inference. In *Proceedings of the AAAI 2014 Workshop on Statistical Relational AI*, 2014.

W. Y. Wang, K. Mazaitis, N. Lao, and W. W. Cohen. Efficient Inference and Learning in a Large Knowledge Base. *Machine Learning*, pages 1–26, 2015.

D. S. Weld, F. Wu, E. Adar, S. Amershi, J. Fogarty, R. Hoffmann, K. Patel, and M. Skinner. Intelligence in Wikipedia. In *AAAI*, volume 8, pages 1609–1614, 2008.

D. C. Wimalasuriya and D. Dou. Ontology-based Information Extraction: An Introduction and a Survey of Current Approaches. *J. Information Science*, 36(3), 2010.

W. E. Winkler. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*. Citeseer, 1999.

T. Winograd. Understanding Natural Language. *Cognitive psychology*, 3(1):1–191, 1972.

L. Yao, S. Riedel, and A. McCallum. Probabilistic Databases of Universal Schema. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 116–121. Association for Computational Linguistics, 2012.

L. Yao, S. Riedel, and A. McCallum. Universal Schema for Entity Type Prediction. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, pages 79–84. ACM, 2013.

J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen. Statsnowball: A Statistical Approach to Extracting Entity Relationships. In *Proceedings of the 18th International Conference on World Wide Web*, pages 101–110. ACM, 2009.