The Dissertation Committee for Lilyana Simeonova Mihalkova
certifies that this is the approved version of the following dissertation:

Learning with Markov Logic Networks: Transfer Learning, Structure Learning, and an Application to Web Query Disambiguation

Committee:

Raymond Mooney, Supervisor

Pedro Domingos

Kristen Grauman

Maytal Saar-Tsechansky

Peter Stone

Learning with Markov Logic Networks: Transfer Learning, Structure Learning, and an Application to Web Query Disambiguation

by

Lilyana Simeonova Mihalkova, B.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2009

To my Mom and Dad.

Acknowledgments

First, I would like to thank my advisor Ray Mooney for his guidance, encouragement, and mentorship during the past six years. I am grateful to Ray for directing me towards statistical relational learning and for our weekly discussions, which significantly shaped the ideas presented in this thesis. I have greatly appreciated Ray's advising style, especially the fact that he fostered an open and friendly research environment and encouraged discussion and independent exploration. Without Ray's balanced mix of guidance and independence, my graduate student experience would not have been as enriching and fulfilling.

I would also like to thank my committee members—Pedro Domingos, Kristen Grauman, Maytal Saar-Tsechansky, and Peter Stone, as well as my past committee member Ben Kuipers. I have enjoyed and appreciated my discussions with them, and their insightful comments and suggestions have helped strengthen the work in this thesis.

Special thanks go to my fellow graduate student Tuyen Huynh for his work on the mapping aspect of TAMAR, as well as for his insightful comments on pretty much all research I have done since then. I would also like to thank past and present members of the Machine Learning group—Misha Bilenko, Razvan Bunescu, David Chen, Ruifang Ge, Sonal Gupta, Tuyen Huynh, Rohit Kate, Joohyun Kim, Prem Melville, Joe Reisinger, and John Wong for coming to my practice talks and for their feedback and comments on paper drafts and presentations. I am also grateful to Matt Richardson for the opportunity to be an intern at MSR under his mentorship. Our discussions helped me deepen my understanding of MLNs, and the internship sparked my interest in Web personalization problems.

I have greatly appreciated the Alchemy software system, on top of which I have built all software for the experiments in this thesis. I am thankful to the Alchemy team, and especially Stanley Kok and Marc Sumner, for making this software available and for their help with it.

Working in the office would not have been nearly as fun without my wonderful office mates. I am especially thankful to Razvan Bunescu and John Wong for their friendship and encouragement when I was most uncertain in myself; and to Sonal Gupta whose friendship and positive and constructive attitude I value immensely.

I am grateful for the opportunity to study at UT Austin, which has provided a wonderful environment in which to learn and grow. I am especially thankful to the technical staff at the Department of Computer Sciences for making everything run smoothly and for helping me with numerous technical questions, as well as to Gloria Ramirez, Lydia Griffith, Stacy Miller, and Katherine Utz for their help with administrative issues.

I have been fortunate to have many wonderful teachers, professors, and mentors who have been extremely helpful and deeply influential at various stages of my education. I would like to thank Mrs. Vanya Hristova—I could not imagine a better and more dedicated teacher than her. I am deeply grateful to Matthew Fike, my first-year-writing-seminar professor at the American University in Bulgaria. I thank Matt for his friendship, mentorship, for teaching me how to become a better writer, and for his great help with the application process.

I would like to thank my professors at Hope College for their dedication and personal involvement in my education. I am especially grateful to Prof. Herb Dershem for teaching the best computer science class I've had (*Languages and Machines*), for leading my first class in machine learning, and for encouraging me to pursue a Ph.D. I would also like to give special thanks to Prof. Ryan McFall for all the valuable things about computer science I learned in his classes (the largest number I have taken with any professor).

On a more personal level, I would like to thank my dear friends for their support throughout my Ph.D. journey: Sabina Amanbayeva, Iva Bozhinova (previously Ivanova), Nikoleta Daskalova, and Valentina Mitova. I am grateful to Larry Schipper for being such a wonderful host father and for introducing me to some of the best aspects of US culture.

This thesis would not have been possible without the unwavering support of my family. I thank them for their belief in me, which means so much. I thank my brother Dimiter Mihalkov for being such a wonderful listener. I thank my grandparents Lilyana and Boris Korsakovi, my grandmother Elisaveta Mihalkova, and my late grandfather Dimiter Mihalkov for being exceptional role models. My parents Stefka and Simeon Mihalkovi denied themselves much in order to provide me with the best possible education. I dedicate this thesis to them.

Finally, I would like to thank my husband Sasha Sherstov for being my tireless cheerleader; for his constructive advice that has helped me overcome so many setbacks; and for the joy and meaning he brings to my life.

Lilyana Mihalkova

August 2009

Austin, TX

Learning with Markov Logic Networks: Transfer Learning, Structure Learning, and an Application to Web Query Disambiguation

Publication No. ____________________

Lilyana Simeonova Mihalkova, Ph.D.
The University of Texas at Austin, 2009

Supervisor: Raymond Mooney

Traditionally, machine learning algorithms assume that training data is provided as a set of independent instances, each of which can be described as a feature vector. In contrast, many domains of interest are inherently *multi-relational,* consisting of entities connected by a rich set of relations. For example, the participants in a social network are linked by friendships, collaborations, and shared interests. Likewise, the users of a search engine are related by searches for similar items and clicks to shared sites. The ability to model and reason about such relations is essential not only because better predictive accuracy is achieved by exploiting this additional information, but also because frequently the goal is to *predict* whether a set of entities are related in a particular way. This thesis falls within the area of Statistical Relational Learning (SRL), which combines ideas from two traditions within artificial intelligence, first-order logic and probabilistic graphical models, to address the challenge of learning from multi-relational data. We build on one particular SRL model, Markov logic networks (MLNs), which consist of a set of weighted first-order-logic formulae and provide a principled way of defining a probability distribution over possible worlds. We develop algorithms for learning of MLN structure both from scratch and by transferring a previously learned model, as well as an application of MLNs to the problem of Web query disambiguation. The ideas we present are unified by two main themes: the need to deal with limited training data and the use of bottom-up learning techniques.

Structure learning, the task of automatically acquiring a set of dependencies among the relations in the domain, is a central problem in SRL. We introduce BUSL, an algorithm for learning MLN structure from scratch that proceeds in a more bottom-up fashion, breaking away from the tradition of top-down learning typical in SRL. Our approach first constructs a novel data structure called a *Markov network template* that is used to restrict the search space for clauses. Our experiments in three relational domains demonstrate that BUSL dramatically reduces the search space for clauses and attains a significantly higher accuracy than a structure learner that follows a top-down approach.

Accurate and efficient structure learning can also be achieved by transferring a model obtained in a *source* domain related to the current *target* domain of interest. We view transfer as a revision task and present an algorithm that diagnoses a source MLN to determine which of its parts transfer directly to the target domain and which need to be updated. This analysis focuses the search for revisions on the incorrect portions of the source structure, thus speeding up learning. Transfer learning is particularly important when target-domain data is limited, such as when data on only a few individuals is available from domains with hundreds of entities connected by a variety of relations. We also address this challenging case and develop a general transfer learning approach that makes effective use of such limited target data in several social network domains.

Finally, we develop an application of MLNs to the problem of Web query disambiguation in a more privacy-aware setting where the only information available about a user is that captured in a short search session of 5–6 previous queries on average. This setting contrasts with previous work that typically assumes the availability of long user-specific search histories. To compensate for the scarcity of user-specific information, our approach exploits the relations between users, search terms, and URLs. We demonstrate the effectiveness of our approach in the presence of noise and show that it outperforms several natural baselines on a large data set collected from the MSN search engine.

Table of Contents

Chapter 1

Introduction

The goal of machine learning is to develop algorithms that allow intelligent systems to acquire knowledge and improve their performance automatically from experience. The typical assumption made by most machine learning algorithms is that training data is provided as a set of instances, where each instance is described as a feature vector, from which the value of a target feature is to be predicted. For example, in a system for evaluating credit card applications, each training instance is a credit card applicant, who is described by a vector of features, such as income, birth date, profession, and address, and the goal is to predict whether or not the applicant is creditworthy. The crucial assumption made by such feature-vector classification algorithms is that the instances are *independent* of each other. Therefore, such algorithms view the data as being represented by a single table that contains a row for each instance and a column for each feature, such that individual rows are independent.

In contrast, many domains of interest are inherently multi-relational, consisting of entities connected by a rich set of relations. For example, the participants in a social network are linked by friendships, collaborations, and shared interests. Likewise, the users of a search engine are related by searches for similar items and clicks to shared sites, which are themselves related by shared topics, keywords, or by linking to one another. The ability to model and reason about such relations is essential not only because better predictive accuracy is achieved by exploiting this additional information, but also because frequently the goal is to *predict* whether a set of entities are related in a particular way. For example, predicting the "friendship" relation allows social networking sites to suggest new friends to their users, whereas by predicting the "interested in" relation, a search engine can customize its results for each user. Algorithms that assume a feature-vector representation cannot be employed for such tasks, in which the data can be viewed as consisting of multiple tables that describe the properties and relations of the same set of entities. For example, in a social networking domain, one table may contain the birth dates, addresses, and other personal information of users; another table may represent friendship relationships by listing pairs of friends; and a third table may represent group memberships by listing person-group pairs.

Statistical relational learning (SRL) (Getoor & Taskar, 2007), a subfield of machine learning, has made great progress in addressing the challenge of learning from such *multi-relational* data by combining ideas from two traditions within artificial intelligence. On the one hand, unlike learning methods that use feature-vector representations, SRL uses the expressiveness of structured languages, such as first-order logic or SQL, to represent the *structure*, which captures the dependencies and regularities among the relations in a domain. On the other hand, SRL borrows ideas from graphical models, such as Bayesian or Markov networks, to impose a probabilistic interpretation over the structure, thus enabling SRL to deal with the noise and uncertainty frequently present in relational domains.

The work in this thesis builds on one particular SRL model, the Markov logic network (MLN) (Richardson & Domingos, 2006). In an MLN, dependencies among the relations are expressed in first-order logic as a set of possibly contradictory formulae, and the weight attached to each formula determines its relative importance in the overall model. Intuitively, each formula in an MLN represents a “rule of thumb” that guides prediction but does not have to be always true. There are several advantages to using MLNs that have motivated our choice of model. First, MLNs are a very expressive and general representation. They are capable of representing all possible probability distributions over a finite number of objects (Richardson & Domingos, 2006) and subsume all SRL representations that can be formed as special cases of first-order logic or probabilistic graphical models (Richardson, 2004). This set includes several widely used models, such as probabilistic relational models (Getoor, Friedman, Koller, & Pfeffer, 2001) and relational Markov networks (Taskar, Abbeel, & Koller, 2002). As a result of this generality, many of the techniques we present are directly applicable to other SRL models. Second, the use of first-order logic to express MLN structure is beneficial because, on the one hand, it allows MLN structure learning techniques to draw inspiration from the rich, decades-long tradition on inductive logic programming (Džeroski & Lavrač, 2001); on the other hand, first-order logic provides an intuitive language in which available background knowledge can be conveniently encoded by human engineers. Third, MLNs come with a well-maintained code base (Kok, Singla, Richardson, & Domingos, 2005), whose availability has allowed us to focus on the novel contributions of this thesis, without having to develop a dedicated framework from scratch.

1.1 Main Themes

The ideas presented in this thesis are unified by two main underlying themes:

1.1.1 Dealing with Limited Training Data

Limited data is a common impediment to successful modeling in machine learning. In this thesis, we have explored two approaches to overcoming this challenge. First, we introduce techniques for transfer across multi-relational domains that enable more accurate learning from limited data. In transfer learning, a model acquired in a source domain is used to aid learning in a target domain that is distinct from the source but related to it. The use of transfer learning techniques may be beneficial in three ways: by giving an initial boost to the learner before any data is observed, by attaining superior performance from less data, and by obtaining more accurate models at the end of learning. Transfer learning in SRL can be viewed as a way of breaking the independent and identically distributed (i. i. d.) assumption commonly made in feature-vector learning. The independence aspect is violated by the fact that in SRL entities can engage in relations; the identically distributed aspect is broken by transfer, which enables learning from training data that follows a different distribution from that of the test.

Second, by focusing on one particular problem, Web query disambiguation, we explore ways in which knowledge about the relations between entities can be

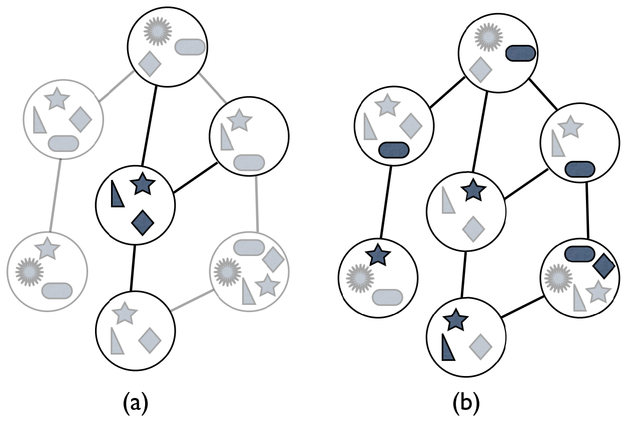


Figure 1.1: Two scenarios of limited data considered in this thesis. Each node represents an entity. The shapes inside a node represent its features, whereas the edges represent relations. In (a), full knowledge about a single node is provided. In (b), there is limited knowledge about the node attributes.

used to compensate for the scarcity of entity-specific feature information. The goal in Web query disambiguation is to determine the intentions of a searcher who enters a potentially ambiguous query. By exploiting the relations between users, we develop an approach which does not depend on extensive user-specific and potentially sensitive personal information. Figure 1.1 illustrates two scenarios of limited data considered in this thesis.

1.1.2 Bottom-Up Learning

Traditionally, learning algorithms in SRL have followed a *top-down* paradigm common in probabilistic graphical model learning where a greedy search through the hypothesis space is conducted by systematically generating a large number of candidates at each iteration, scoring them according to a probabilistic measure, and keeping the most promising ones, from which new candidates are generated at the next iteration (e.g., Heckerman, 1995). In contrast, the techniques introduced in this thesis follow a *bottom-up* philosophy, and take a more data-driven approach, whereby a close analysis of the available data motivates a smaller number of more promising candidate hypotheses. Because of this, bottom-up techniques are usually faster to train. Bottom-up learning is also motivated by the observation that it frequently leads to more accurate models because the guidance from the data prevents learning from being trapped in local maxima. We have explored two aspects of this theme: in MLN structure learning from scratch, where our approach first summarizes useful features in a novel data structure that then guides the search for structures; and in transfer of MLN structure, where the data is used to diagnose a source model, thus narrowing down the search for corrections. The difference between the top-down and bottom-up paradigms is illustrated in Figure 1.2.

1.2 Thesis Contributions

The goal of this thesis is to address several aspects of learning with MLNs: structure learning, transfer learning, and an application to Web query disambiguation, as we describe next in more detail.

1.2.1 Structure Learning

A central problem in SRL is *structure learning*, the task of automatically acquiring a set of dependencies among the relations in the domain. We introduce a novel approach for learning MLN structure from scratch called BUSL for Bottom-

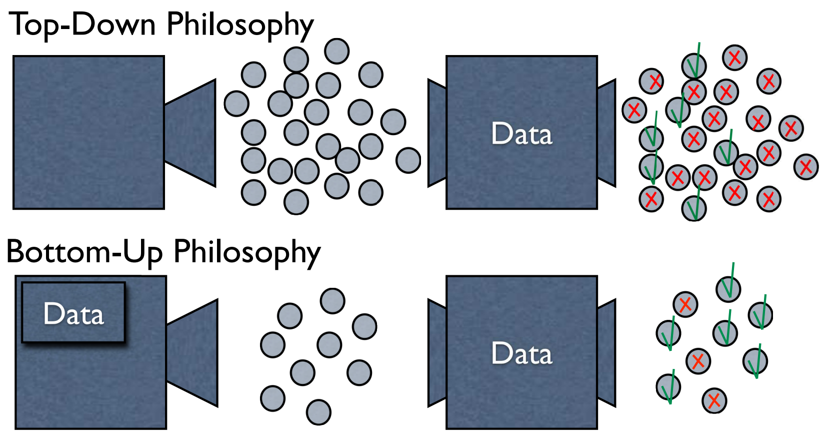


Figure 1.2: An illustration of top-down versus bottom-up learning. In top-down learning, a large number of candidate hypotheses (the gray circles) are generated, and the data is used only to evaluate these candidates. In bottom-up learning, the data is used also to drive the generation of hypotheses; as a result, a smaller number of the more promising candidates is generated.

Up Structure Learning. Our approach breaks away from the top-down paradigm and instead proceeds in a more bottom-up fashion by first constructing a *Markov network template*, a variablized Markov network, whose nodes consist of chains of one or more literals and serve as clause building blocks. The Markov network template is used to restrict the search space for clauses by requiring that all literals in a clause be part of a clique in the template. This restriction is motivated by the observation that the clauses in an MLN define functions over the cliques of the Markov network that is obtained by grounding the MLN for a particular domain. Our experiments in three real relational domains demonstrate that this approach dramatically reduces the search space for clauses and attains a significantly higher accuracy than a structure learner that follows a top-down approach (Kok & Domingos, 2005).

1.2.2 Transfer Learning

Accurate and efficient structure learning can also be achieved by transferring a source model obtained in a *source* domain related to the current *target* domain of interest. For example, because human interactions tend to be similar across contexts, a model learned in a domain on social interactions in the movie business is likely to be effective in a domain about social interactions in academia. We view transfer as a revision task and present an algorithm that diagnoses a source MLN to determine which of its parts transfer directly to the target domain and which need to be updated. This analysis focuses the search for revisions on the incorrect portions of the source structure, thus speeding up learning. We also demonstrate that when this revision technique is incorporated in an integrated transfer system that first maps the source knowledge to the target domain and then revises it, improvements in the accuracy of learning over learning from scratch can also be obtained.

Transfer learning is particularly important when target-domain data is limited, such as when data on only a few individuals is available from domains with hundreds of entities connected by a variety of relations. We address this case, in which learning from scratch is infeasible, and develop SR2LR, a general transfer learning approach that makes effective use of such limited target data in several social network domains.

1.2.3 Web Query Disambiguation

We develop an application of MLNs to the problem of Web query disambiguation in a more privacy-aware setting where the only information available about a user is that captured in a short search session of 5 to 6 previous queries on average. This setting contrasts with previous work that typically assumes the availability of long user-specific search histories, and is of significant practical importance for users who want a personalized experience but are wary of having long histories of their searches be recorded by the search engine. To compensate for the scarcity of user-specific information, our approach exploits the relations between users, search terms, and URLs, and uses a hand-coded structure, over which weights are learned in an online fashion. We demonstrate the effectiveness of our approach in the presence of noise and show that it outperforms several natural baselines on a large data set collected from the MSN search engine.

1.3 Thesis Roadmap

In the next chapter, we start with a discussion of the background on which the contributions of this thesis build. Chapter 3 discusses our work in transfer learning by describing two algorithms—rTAMAR and SR2LR. Chapter 4 describes our algorithm for learning the structure of MLNs from scratch in a bottom-up way. In Chapter 5, we develop an approach to the problem of Web query disambiguation and demonstrate how relational information can be exploited to compensate for a limitation on the amount of user-specific data. Chapter 6 describes future directions, and Chapter 7 concludes.

Chapter 2

Background

The work in this thesis is in the area of Statistical Relational Learning (SRL) (Getoor & Taskar, 2007), which builds upon two major traditions within artificial intelligence—logical models, and probabilistic graphical models. In this chapter we give a brief overview of the necessary background in these two areas and then describe in detail Markov logic networks (Richardson & Domingos, 2006), the specific SRL model upon which we build. Later chapters will introduce related work specific to their content.

2.1 First-Order Logic

First-order logic provides an expressive language for describing the features and relations that hold in an environment. It distinguishes among four types of symbols—constants, variables, predicates, and functions (Russell & Norvig, 2003). Constants describe the objects in a domain and can have types. For example, a domain may contain the constants `jack` and `jill` of type person and `male` and `female` of type gender. Variables act as placeholders to allow for quantification. Predicates represent relations in the domain, such as `WorkedFor`. Function symbols represent functions over tuples of objects. The arity of a predicate or a function is defined as the number of arguments it takes. These arguments can also be typed, thus restricting the type of constant that can be used. We will denote constants by strings starting with lower-case letters (i.e. `jill`), variables by single upper-case letters (i.e A, B), and predicates by strings starting with upper-case letters (i.e. `WorkedFor`). Sets of variables will be denoted with bold upper-case letters (i.e. $\mathbf{A}, \mathbf{B}$).

Example 2.1.1. As a running example, we will use the following simplified version of one of our test domains. The domain contains facts about individuals in the movie business, describing their profession (`Actor(A)` or `Director(A)`), their relationships, and the movies on which they have worked. The `WorkedFor(A, B)` predicate specifies that person A worked on a movie under the supervision of director B, whereas the `Credits(T, A)` predicate specifies that individual A appeared in the credits of movie T. Here A, B, and T are variables. `Actor` and `Director` each have one argument of type person; `WorkedFor` has two arguments of type person; and `Credits` has two arguments where the first one is of type movieTitle and the second one is of type person. Our example domain has the constants `brando` and `coppola` of type person, and `godFather` of type movieTitle.

A term is a constant, a variable, or a function that is applied to terms. Ground terms contain no variables. An atom is a predicate applied to terms. A positive literal is an atom, and a negative literal is a negated atom. We will use the term **gliteral** to refer to a ground literal, i.e. one containing only constants, and

vliteral to refer to a literal that contains only variables. A clause is a disjunction of positive and negative literals. Ground clauses contain only gliterals. The length of a clause is the number of literals in the disjunction. A definite clause is a clause with exactly one positive literal, called the head, whereas the negative literals compose the body. A Horn clause is a clause with at most one positive literal. A world is an assignment of truth values to all possible gliterals in a domain. If the closed-world assumption is made, only the true gliterals need to be listed; under this assumption all unlisted gliterals are assumed to be false. For the remainder of this document, we will make the closed-world assumption, unless otherwise specified.

Example 2.1.2. For example, `WorkedFor(A, B)` is a vliteral, while `WorkedFor (brando, coppola)` is a gliteral. The following clause is definite because it contains exactly one positive literal:

$$\text{Credits(T,B)} \vee \neg\, \text{Credits(T,A)} \vee \neg\, \text{WorkedFor(A, B)}.$$

Using the fact that $q \vee \neg p$ is logically equivalent to $p \Rightarrow q$, we can rewrite this clause in a more human-readable way, without modifying its meaning, as follows:

$$\text{Credits(T, A)} \wedge \text{WorkedFor(A, B)} \Rightarrow \text{Credits(T,B)}.$$

Note that every definite clause of length at least 2 can be rewritten as a conjunction of positive literals that serve as the premises (the body) and a conclusion consisting of a single positive literal (the head).

One possible grounding of the above clause is:

$$\text{Credits(godFather, brando)} \wedge \text{WorkedFor(brando, coppola)} \Rightarrow$$
$$\text{Credits(godFather,coppola).}$$

In fact, we can rewrite any clause as an implication. Consider, for example, the following clause, which is neither Horn, nor definite because it contains more than one positive literal:

$$\text{Actor(A)} \vee \neg\, \text{Credits(T, A)} \vee \text{Director(A)}$$

This clause can be rewritten as an implication in several ways, depending on what literal we would like to serve as the conclusion:

$$\neg\text{Actor(A)} \wedge \text{Credits(T, A)} \Rightarrow \text{Director(A)}$$

$$\text{Credits(T, A)} \wedge \neg\, \text{Director(A)} \Rightarrow \text{Actor(A)}$$

$$\neg\, \text{Actor(A)} \wedge \neg\, \text{Director(A)} \Rightarrow \neg\, \text{Credits(T, A)}$$

We will call the literals to the left of the implication **premises** or **antecedents**. The literal on the right of the implication will be called the **conclusion**. These implication rewrites will be helpful in Section 3.2.

2.2 Inductive Logic Programming

Inductive logic programming (ILP) is an area within machine learning that studies algorithms for learning sets of first-order clauses (Lavrač & Džeroski, 1994).

Usually, the task is to learn rules for a particular target predicate, such as `WorkedFor`, given background knowledge. This background knowledge may consist either of general clauses, or, more commonly, of a list of the true gliterals of all predicates in the domain except the target predicate. The negative and positive examples are provided by the true and false gliterals of the target predicate (i.e. in our case `WorkedFor`). The form learned rules can take is frequently restricted by demanding that they be definite clauses (e.g., Richards & Mooney, 1995), or by allowing the user to impose some other declarative bias (e.g., De Raedt & Dehaspe, 1997). By performing techniques such as resolution on the learned clauses, new examples can be classified as positive or negative.

2.2.1 Top-Down ILP

Top-down ILP algorithms (e.g., Quinlan, 1990; De Raedt & Dehaspe, 1997) search the hypothesis space by considering, at each iteration, all valid refinements to a current set of candidate hypotheses. These candidates are then evaluated based on how well they cover positive examples and exclude negatives, a set of well-performing ones is greedily selected, and the process continues with the next iteration. In addition to classification accuracy, several other heuristics for scoring, or evaluating, candidates have been used. For example, FOIL uses an information theoretic measure of the information gained by adding a literal to a candidate clause (Quinlan, 1990). CLAUDIEN uses a measure that takes into account the length of the clause (De Raedt & Dehaspe, 1997). In summary, top-down ILP techniques use the data only to evaluate candidate hypotheses but not to suggest ways for forming new candidates.

2.2.2 Bottom-Up ILP

Bottom-up ILP algorithms start with the most specific hypothesis and proceed to generalize it until no further generalizations are possible without covering some negative examples (Lavrač & Džeroski, 1994). For example, the initial hypothesis may be a set of rules where each rule's premises are simply a conjunction of the true gliterals in the background knowledge, and the conclusion is one of the positive examples. One way of generalizing this initial set of clauses is via the technique of *least general generalization* (LGG) (Plotkin, 1970), which can be intuitively understood as the most cautious, or conservative, generalization. The technique of LGG is appealing also because it provides a principled way of dealing with functions. One popular ILP system that uses LGG is GOLEM (Muggleton & Feng, 1992). LGG has also been used by Thomas (2003) to develop an algorithm that extracts information from hypertext documents.

An alternative method for bottom-up ILP is known as inverse resolution (Lavrač & Džeroski, 1994), in which the basic idea is to start from a positive example in the data and attempt to construct rules from which the example can be derived using resolution. The LOGAN-H algorithm (Arias, Khardon, & Maloberti, 2007) generates clause candidates in a similar way, but rather than proposing new clauses based on the positive examples, it uses the negative examples. Given a negative example, it generates the set of all Horn clauses, such that the antecedents consist of all true statements in the example, and the conclusion is a fact that is *not* true in the given example. LOGAN-H then uses a novel generalization procedure that modifies not just the antecedents of a Horn clause, but also the set of possible conclusions. Positive examples are used only for removing clauses with incorrect conclusions.

In summary, bottom-up ILP algorithms take stronger guidance from the data, which is also used to *propose* clause candidates. This is in contrast with top-down algorithms, which use the data only to evaluate candidate clauses.

2.2.3 Hybrid Approaches

Hybrid approaches, (e.g., Zelle, Mooney, & Konvisser, 1994; Muggleton, 1995), aim to exploit the strengths of top-down and bottom-up techniques while avoiding their weaknesses. Because bottom-up techniques generalize from single examples, they are very sensitive to outliers and noise in the training data; however, because many bottom-up techniques employ LGGs, they are better-suited for handling functions. Similarly, top-down techniques can better make use of general background knowledge to evaluate their hypotheses, but the greedy search through the hypothesis space can lead to long training times.

For example, Zelle et al. (1994) present an approach, CHILLIN, that successfully improves accuracy over both a purely top-down and a purely bottom-up learner by combining ideas from these two paradigms. CHILLIN uses LGGs to form initial clauses and refines them further by searching for additional antecedents in a top-down way, as well as inventing new predicates that are necessary in order to express the target concept concisely.

Relational pathfinding (RPF), developed by Richards and Mooney (1992), is another hybrid approach to clausal discovery. RPF views the relational domain as a hypergraph G in which the constants are the vertices and a set of constants are connected by a hyperedge if they appear together in a true gliteral. Intuitively, RPF forms definite clauses in which the head is a particular true gliteral of the target predicate, and the body consists of gliterals that define a path in the relational graph G between the constants in that gliteral. These clauses are then variablized. More specifically, RPF searches G for an alternate path of length at least 2 between a set of constants $\{c_1, \ldots, c_a\}$ connected by a hyperedge, where a is the arity of the target predicate. If such a path is found, it is transformed into a clause as follows. First, a *negative* literal is created for each predicate that labels a hyperedge in the path and is grounded with the constants connected by this hyperedge. In addition, a *positive* literal is constructed in this way for the hyperedge connecting $\{c_1, \ldots, c_a\}$. The resulting clause is a disjunction of these literals with constants replaced by variables. This is the bottom-up part of the process. Hill-climbing search, which proceeds in a top-down fashion, is used to further improve the clauses by possibly adding unary predicates.

Example 2.2.1. Suppose Figure 2.1 lists all true facts in the domain. Figure 2.2 shows the relational graph for this domain, in which all predicates are of arity at most two. The highlighted edges form an alternative path between `brando` and `coppola`, from which we construct the clause:

WorkedFor(brando,coppola) $\vee\neg$Credits(godFather,brando) $\vee\neg$Credits(godFather,coppola).

Director(coppola) Actor(brando)
Credits(godFather, brando) Credits(godFather, coppola)
Credits(rainMaker, coppola) WorkedFor(brando, coppola)

Figure 2.1: Example relational database

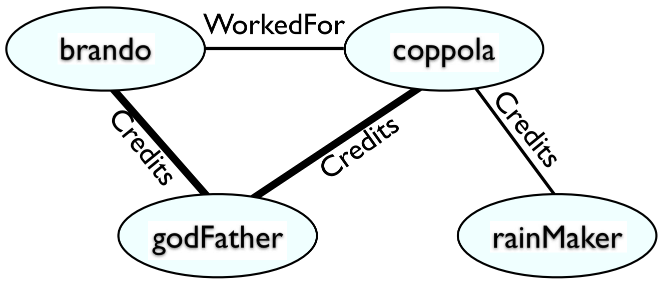


Figure 2.2: Example of a relational graph

After variablizing, this clause becomes:

$$\text{WorkedFor(A,B)} \vee \neg \text{Credits(T,A)} \vee \neg \text{Credits(T,B)}.$$

This can be rewritten as

$$\text{Credits(T,A)} \wedge \text{Credits(T,B)} \Rightarrow \text{WorkedFor(A,B)}.$$

Hill-climbing search might lead to the addition of `Actor(A)` and `Director(B)` to the conjunction in the antecedents.

2.2.4 Revision of Logic Programs

The ILP algorithms discussed so far all learn from scratch. Sometimes, however, an initial, somewhat incorrect, first-order logic theory is provided, along with training data, and the task is to revise the theory so that it fits the training data by modifying it as little as possible. This is the problem addressed by Richards and Mooney (1995). The resulting system, FORTE, can be viewed as a hybrid revision algorithm. FORTE is a top-down learner in that it uses hill-climbing search to improve the provided theory. However, rather than attempting all possible refinements to the provided clauses, FORTE starts in a bottom-up fashion and focuses its search by first diagnosing the possible sources of errors in the provided theory. It does this by attempting to prove positive examples and observing where the clauses fail. These points of failure are marked as revision points and are the only places in the original theory where attempts for improvements are made.

More recently, Goldsmith and Sloan (2005) present revision algorithms for restricted classes of Horn clauses. They give an algorithm for the case of depth-one acyclic Horn clauses in which variables that occur as a head in a clause do not appear in the body of any other clause. A second algorithm deals with the restricted case of Horn clauses with unique heads. The introduction of these subclasses of Horn clauses, allows the authors to give theoretical guarantees of the efficiency of their algorithms.

As we will discuss in Section 3.1.4, revision algorithms can be used for transfer learning where the initial first-order logic theory is learned in a previous domain, rather than being provided by a human.

All the approaches discussed in Section 2.2 result in the construction of first-order theories. Even though this representation is highly expressive, it is not well-suited to modeling in uncertain domains and cannot provide estimates of the probability that a certain fact is true. Next, we turn to an overview of probabilistic graphical models, which provide an important step towards modeling uncertainty.

2.3 Probabilistic Graphical Models

Probabilistic graphical models provide a compact way of representing a joint probability distribution over sets of variables. Assuming that each variable can take on at most v values, any joint probability distribution can be expressed by listing the probability for every possible combination of assignments of values to the variables. If the total number of variables is n, this would require one to specify v^n parameters. Probabilistic graphical models take advantage of the observation that frequently a given variable is directly dependent on only a small subset of the variables, and this subset renders it conditionally independent of the rest. Thus, in a complete listing of probabilities for all possible value combinations, many of the parameters will have the same value. Probabilistic graphical models avoid this redundancy by explicitly modeling the conditional independencies in the domain. The variables are represented as nodes in a graph and the edges indicate dependencies among the variables. Probabilities are computed via a set of functions defined over the graph. For example, Bayesian networks (Pearl, 1988) are a popular model represented as a directed acyclic graph, in which the joint probability is computed using a set of conditional probability functions, one for each node in the graph, that specify the probability of that node taking a particular value given the values of its parents in the graph. Another popular probabilistic graphical model are Markov networks (Pearl, 1988), which, in contrast to Bayesian networks, are represented by undirected graphs and are therefore easier to learn because one does not need to ensure that the resulting graph is acyclic. Next, we describe in detail Markov networks because they will be important for understanding the work in this thesis.

2.3.1 Markov Networks

A Markov network (Pearl, 1988), also known as a Markov random field (Della Pietra, Della Pietra, & Lafferty, 1997), is represented as an undirected graph G in which there is a vertex for each variable in the domain. The interpretation of G is that each variable X is conditionally independent of all other variables, given its immediate neighbors. Because of its importance, the set of immediate neighbors of X is called a *Markov Blanket* of X and we will denote it with MB_X.

The probability distribution defined by a Markov network is described by a set of nonnegative functions $g_i(\mathbf{C_i})$ where $\mathbf{C_i}$ consists of the variables in the i-th maximal clique of G. The probability of assigning particular values $\mathbf{x}$ to the set of variables $\mathbf{X}$ in G (with the cliques having values $\mathbf{c_i}$) is:

$$P(\mathbf{X}=\mathbf{x}) = \frac{\prod_i g_i(\mathbf{c_i})}{\sum_{\mathbf{y}} \prod_i g_i(\mathbf{c_i})} \tag{2.1}$$

The function in the denominator, known as the *partition function*, simply sums over the values of the numerator for all possible value assignments to the variables and serves as a normalizing term. Intuitively, it is possible to represent a probability distribution that preserves the conditional independencies captured by G as a product of functions over only the cliques of G because a variable only directly influences its neighboring variables. This intuition has been formalized as the Hammersley Clifford Theorem (Hammersley & Clifford, 1971), which states that if P is a strictly positive probability distribution (i.e. every event has some chance of happening), then it can be expressed as a product of functions over the cliques of a graph G_P if and only if every conditional independence implied by the structure of G_P exists in P.

Markov networks are most commonly represented as log-linear models where the functions $g_i(\mathbf{C_i})$ take the form $\exp(\lambda_i f_i(\mathbf{C_i}))$. The λ_i-s are called weights, and the f_i-s are called features. With this formulation, Equation 2.1 can be rewritten as follows:

$$P(\mathbf{X} = \mathbf{x}) = \frac{\exp\left(\sum_i \lambda_i f_i(\mathbf{C_i})\right)}{\sum_{\mathbf{y}} \exp\left(\sum_i \lambda_i f_i(\mathbf{C_i})\right)} \tag{2.2}$$

Apart from their convenience, log-linear models are desirable also because it can be shown that if such a model is used, optimizing the weights in order to maximize the data likelihood, leads to the model with the highest entropy (Berger, 1996; Della Pietra et al., 1997).

2.3.2 Learning of Markov Networks

If the features are given, one effective way of learning the weights is by using gradient descent because, for fixed features, optimization of the weights is over a convex space (Della Pietra et al., 1997). One common approach to learning the features of Markov networks, also known as *structure learning*, is by proceeding in iterations where in each iteration the feature that gives the best improvement in data fit is greedily added. For example, Della Pietra et al. (1997) choose the feature that gives the largest decrease in Kullback-Leibler divergence between the empirical distribution of the data and the distribution represented by the current model. It is also common to add a term that penalizes complex models (e.g., Lee, Ganapathi, & Koller, 2007). These types of approaches are *feature-centric* in that they focus on selecting the features that give the best immediate advantage, without considering the underlying graph structure and the implied conditional independencies among the variables.

An alternative approach to learning Markov networks is to proceed in a *graph-centric* way by first focusing on establishing a graph structure that asserts the existing conditional independencies among the variables. One such algorithm, which we will use in Chapter 4, is the Grow-Shrink Markov Network (GSMN) algorithm by Bromberg, Margaritis, and Honavar (2006). For each variable X, GSMN goes through two stages—grow and shrink. In the grow phase, the algorithm incrementally constructs the Markov blanket, MB_X, of each variable X. Initially MB_X is empty. The algorithm goes through all other nodes and at each iteration, uses the χ^2 test to determine whether X and Y are conditionally independent given MB_X, where Y is the current potential addition to MB_X. If the two variables are not conditionally independent, Y is added to MB_X. In the shrink phase, GSMN goes through each node $Y \in \text{MB}_X$ and attempts to remove it by testing whether X and Y are conditionally independent given $\text{MB}_X \setminus Y$. After going through the grow and shrink stages for each node, GSMN enters a collaboration phase in which the algorithm ensures that for all pairs of nodes X and Y, if $Y \in \text{MB}_X$, then $X \in \text{MB}_Y$.

Graph-centric algorithms for learning of other probabilistic graphical models include SGS and PC (Spirtes, Glymour, & Scheines, 2001) and the algorithm of Margaritis and Thrun (2000), all of which learn Bayesian networks based on independence tests among the variables, as well as the approach of Abbeel, Koller, and Ng (2006), which constructs Markov blankets using conditional entropy.

Probabilistic graphical models can effectively represent probability distributions over a set of variables. However, they can capture dependencies only over a fixed set of (propositional) variables and cannot concisely model generally valid relationships that hold over large groups of objects. Next, we turn to a short description of statistical relational learning, which aims at overcoming this problem by incorporating ideas from first-order logic, while still maintaining the advantages of probabilistic graphical models.

2.4 Statistical Relational Learning

Statistical relational learning (SRL) (Getoor & Taskar, 2007) combines ideas from first-order logic and probabilistic graphical models to develop learning models and algorithms capable of representing complex relationships among entities in uncertain domains. As opposed to traditional classification where it is assumed that each testing instance is independent of the rest, SRL is best suited to situations in which the entities to be classified are interrelated and the label of one affects the classification of the remaining ones in some non-trivial way. Moreover, SRL addresses the case where learning occurs from multi-relational data, and thus training instances have varying numbers of entities and relations.

Some popular SRL models include probabilistic relational models (PRMs) (Getoor et al., 2001) and Bayesian logic programs (BLPs) (Kersting & De Raedt, 2001), which are both relational analogs to Bayesian networks; as well as relational Markov networks (RMNs) (Taskar et al., 2002) and Markov logic networks (Richardson & Domingos, 2006), which are relational analogs to Markov networks.

In the remainder of this section, we will describe in detail Markov logic networks, which are the SRL model on which this thesis builds. The choice of this model is motivated by the fact that it is highly expressive and subsumes all SRL models that can be formed as special cases of first-order logic and probabilistic graphical models (Richardson, 2004).

2.4.1 Markov Logic Networks

Markov logic networks (MLNs), introduced by Richardson and Domingos (2006), consist of a set of first-order clauses, each of which has an associated weight. MLNs can be viewed as relational analogs to Markov networks whose features are expressed in first-order logic. In this way MLNs combine the advantages of first-order logic with those of probabilistic graphical models while avoiding the drawbacks of the two representations. In particular, the expressive power of first-order logic enables MLNs to represent complex general relationships and to reason about variable numbers of entities using the same model. On the other hand, because the first-order logic features are embedded in the framework of probabilistic graphical models, MLNs avoid the brittleness of pure first-order logic by making worlds that violate some of the clauses less likely but not altogether impossible.

Next, we provide a formal description of MLNs. Let $\mathbf{X}$ be the set of all propositions describing a world (i.e. these are all possible gliterals that can be

0.7	Actor(A) $\Rightarrow$ ¬Director(A)
1.2	Director(A) $\Rightarrow$ ¬WorkedFor(A, B)
1.4	Credits(T, A) $\wedge$ WorkedFor(A, B) $\Rightarrow$ Credits(T,B)

Figure 2.3: Simple MLN for the sample domain

formed by grounding the predicates with the constants in the domain), $\mathcal{F}$ be the set of all first-order clauses in the MLN, and w_i be the weight associated with clause $f_i \in \mathcal{F}$. Let $\mathcal{G}_{f_i}$ be the set of all possible groundings of clause f_i with the constants in the domain. Then, the probability of a particular truth assignment $\mathbf{x}$ to $\mathbf{X}$ is given by the formula (Richardson & Domingos, 2006):

$$P(\mathbf{X} = \mathbf{x}) = \frac{\exp\left(\sum_{f_i \in \mathcal{F}} w_i \sum_{g \in \mathcal{G}_{f_i}} g(\mathbf{x})\right)}{\sum_{\mathbf{y}} \exp\left(\sum_{f_i \in \mathcal{F}} w_i \sum_{g \in \mathcal{G}_{f_i}} g(\mathbf{y})\right)} \tag{2.3}$$

The value of $g(\mathbf{x})$ is either 1 or 0, depending on whether g is satisfied. Thus the quantity $\sum_{g \in \mathcal{G}_{f_i}} g(\mathbf{x})$ simply counts the number of groundings of f_i that are true given the current truth assignment to $\mathbf{X}$. The denominator is the normalizing partition function. Intuitively w_i determines how much less likely is a world in which a grounding of f_i is not satisfied than one in which it is satisfied. The first-order clauses are commonly referred to as *structure*. Figure 2.3 shows a simple MLN that provides an example for our simplified movie domain. Note that the first-order formulas do not have to have any particular form, e.g., they are not restricted to being definite.

To perform inference over a given MLN, one needs to ground it into its corresponding Markov network. As described by Richardson and Domingos (2006), this is done as follows. First, all possible gliterals in the domain are formed, and they serve as the nodes in the Markov network. The edges are determined by the groundings of the first-order clauses: gliterals that participate together in a grounding of a clause, are connected by an edge. Thus, nodes that appear together in a ground clause form cliques. For example, Figure 2.4 shows the ground Markov network corresponding to the MLN in Figure 2.3 using the constants *coppola* and *brando* of type person and *godFather* of type movieTitle. It is also useful to note the similarity between equation 2.3 and equation 2.2. MLNs can be considered as a concise and general way of specifying Markov networks in which there is a feature for each grounding of each clause, and features that correspond to the same unground clause have the same weight.

One technique that can be used to perform inference over the ground Markov network is Gibbs sampling (Richardson & Domingos, 2006). The goal of sampling is to compute the probability that each of a set of query gliterals is true, given the values of the remaining gliterals as evidence. Gibbs sampling starts by assigning a truth value to each query gliteral. This can be done either randomly or by using a weighted satisfiability solver such as MaxWalksat (Kautz, Selman, & Jiang, 1997) that initializes the truth values to maximize the sum of the weights. It then proceeds in rounds to re-sample a value for gliteral X, given the truth values of its Markov blanket MB_X (i.e. the variables with which it participates in ground clauses), using the following formula to calculate the probability that X takes on a particular value x.

$$P(X = x|\text{MB}_X = \mathbf{m}) = \frac{e^{S_X(x,\mathbf{m})}}{e^{S_X(0,\mathbf{m})} + e^{S_X(1,\mathbf{m})}}. \tag{2.4}$$

Here, $S_X(x, \mathbf{m}) = \sum_{g_i \in \mathcal{G}_X} w_i g_i(X = x, \text{MB}_X = \mathbf{m})$, where $\mathcal{G}_X$ is the set of ground clauses in which X appears and $\mathbf{m}$ is the current truth assignment to MB_X. Efficiency can be improved by including only the query gliterals and those in the Markov blanket of a gliteral with an unknown value, rather than fully grounding the MLN (Richardson & Domingos, 2006).

An alternative inference approach is MC-SAT that has been shown to outperform Gibbs sampling in both speed and the accuracy of the returned probability estimates (Poon & Domingos, 2006). In addition, meta-inference techniques have been developed that improve either memory usage (e.g., Singla & Domingos, 2006) or inference time (e.g., Mihalkova & Richardson, 2009).

2.4.2 Learning Of Markov Logic Networks

Learning of MLNs can proceed in two ways, discriminatively or generatively. In discriminative training, one or more predicates whose values will be unknown at test-time are designated as target predicates, and learning optimizes the performance with respect to them, assuming that values for the remaining predicates will be given. In generative training, all predicates are treated equally. Roughly speaking, discriminative training is appropriate when it is known ahead of time what kind of predictions will need to be performed with the learned model, whereas generative training is appropriate when it is not known ahead of time how the model will be used so that the learned model needs to capture as many aspects of a domain as possible. Detailed studies of the relative advantages of the two styles of training are available (e.g., Liang & Jordan, 2008).

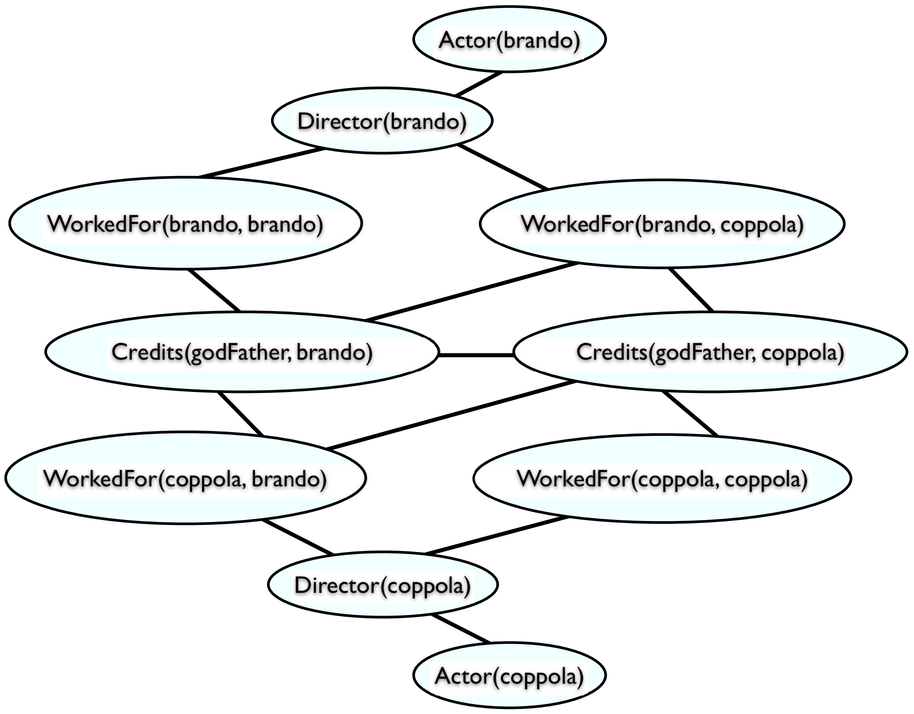

Actor(brando)
Director(brando)
WorkedFor(brando, brando)
WorkedFor(brando, coppola)
Credits(godFather, brando)
Credits(godFather, coppola)
WorkedFor(coppola, brando)
WorkedFor(coppola, coppola)
Director(coppola)
Actor(coppola)

Figure 2.4: Result of grounding the sample MLN

As with Markov networks, there are two parts to learning an MLN: the weights and the structure.

Weight Learning: Richardson and Domingos (2006) propose performing generative weight learning for a fixed set of clauses using L-BFGS (D. C. Liu & Nocedal, 1989), a second-order optimization procedure, to optimize the pseudo log-likelihood (Besag, 1986). Several approaches have been proposed for discriminative learning, where the conditional log-likelihood is optimized instead. The earliest, by Singla and Domingos (2005), follows a voted-perceptron-like approach (Collins, 2002), where the gradient of the conditional log-likelihood with respect to the weight of a given clause C_i is computed as the difference between the number of true groundings of C_i in the data and the expected number of true groundings of C_i according to the current weights. Calculating this expectation requires inference over the learned model, and Singla and Domingos (2005) used the number of true groundings of C_i in the most likely assignment of truth values to approximate it. In later work, Lowd and Domingos (2007) considered calculating the expectation by performing a few steps MC-SAT inference, thus obtaining a contrastive-divergence-like approach (Hinton, 2000). In Chapter 5, we will use this algorithm, which we adapt for online learning. Lowd and Domingos (2007) also studied more sophisticated techniques, such as the preconditioned scaled conjugate gradient algorithm that uses the inverse diagonal Hessian matrix as a preconditioner. Huynh and Mooney (2008) introduce a weight-learning technique that targets the case of MLNs containing only non-recursive clauses.[1] Because of this special assumption on the structure of the model, their approach can perform exact inference when calculating the expected number of true groundings of a clause; a second novelty is the use of L_1 regularization to obtain sparser models in which many clauses have weight 0. Recently, Huynh and Mooney (2009) have introduced a discriminative learner that maximizes the margin between negative and positive gliterals in the training data.

Structure Learning: Structure learning is a highly computationally intensive process. The first MLN structure learner, due to Kok and Domingos (2005), proceeds

[1]Non-recursive clauses mention a target predicate at most once.

in a top-down fashion, employing either beam search or shortest-first search. We will discuss and compare to the beam search version, which we will call KD after its authors. The shortest-first search constructs candidates in the same way but conducts a more complete search, which, however, requires longer training times. KD performs several iterations of beam search, and after each iteration adds to the MLN the best clause found. Clauses are evaluated using a weighted pseudo log-likelihood measure (WPLL)(Kok & Domingos, 2005), an extension of pseudo log-likelihood (Besag, 1986), that sums over the log-likelihood of each node given its Markov blanket, weighting it appropriately to ensure that predicates with many gliterals do not dominate the result. The beam search in each iteration starts from all single-vliteral clauses. It generates candidates by adding a vliteral in each possible way to the initial clauses, keeps the best $beamSize$ clauses, from which it generates new candidates by performing all possible vliteral additions, keeps the best $beamSize$ and continues in this way until candidates stop improving the WPLL. At this point, the best candidate found is added to the MLN, and a new beam search iteration begins. Weights need to be learned for a given structure before its WPLL can be computed. KD has been empirically shown to outperform an impressive number of competitive baselines (Kok & Domingos, 2005). In particular, it performed better than several popular inductive logic programming algorithms and also outperformed purely probabilistic methods.

At the time of writing of this manuscript, Kok and Domingos (2009) have just introduced LHL, a new algorithm for MLN structure learning, which, like BUSL, presented in Chapter 4, embraces a bottom-up perspective. LHL performs relational pathfinding (Richards & Mooney, 1992) on a *lifted hypergraph* constructed by clustering the constants in the data; lifting the hypergraph allows LHL to search for longer paths than BUSL in a reasonable amount of time. We discuss the performance of LHL in Chapter 4.

The above two algorithms take a generative approach. Discriminative structure learners have also been introduced. Huynh and Mooney (2008) use ALEPH (Srinivasan, 2001), a bottom-up ILP system, to learn non-recursive clauses. They found that for molecular biology domains in which the clauses serve primarily to describe complex molecules and tend to be very long, learners such as ALEPH that have been especially designed to deal with such challenges learn more accurate structure. Biba, Ferilli, and Esposito (2008) have introduced a discriminative structure learning algorithm based on iterated local search.

Chapter 3

Transfer Learning with MLNs

Traditional machine learning algorithms operate under the assumption that learning for each new task starts from scratch, thus disregarding any knowledge gained previously. In related domains, this *tabula rasa* approach would waste data and computer time to develop hypotheses that could have been recovered faster from previously acquired knowledge. Transfer learning, also known as learning to learn (Thrun & Pratt, 1998) or domain adaptation (Blitzer, McDonald, & Pereira, 2006), addresses the problem of how to leverage knowledge from related *source* domains in order to improve the efficiency and accuracy of learning in a new *target* domain (Silver et al., 2005; Banerjee et al., 2006; Taylor, Fern, & Driessens, 2008). Transfer learning is also one of the most effective techniques for enabling learning in situations when an adequate amount of training data for the task of interest is not available.

In this chapter, we present two approaches for transfer of MLN structure. The first technique improves the speed and accuracy of learning by operating under the assumption that a substantial amount of data for the target task is provided. The second technique addresses the challenging scenario when target-domain data is severely limited. Unlike most work in transfer learning, our contributions address the setting of multi-relational data and do not assume that the source and target domains use the same representation. Before presenting our contributions, we review related work.

3.1 Related Work

Transfer learning algorithms have been demonstrated to improve learning in a variety of settings. In this section, we discuss related work to provide a glimpse of the numerous transfer algorithms that have been developed.

3.1.1 Multi-task Transfer Learning

Transfer learning has been studied in two main settings. In the multi-task setting, the algorithm is presented with all domains simultaneously during training and thus can build common structure of the learned models. For example, Caruana (1997) trained neural networks with a shared hidden layer on two or more tasks simultaneously. A related approach is used by Niculescu-Mizil and Caruana (2005, 2007) for simultaneous training of Bayesian networks. In a similar setting, Ando and Zhang (2005) perform optimization over a set of tasks simultaneously to find an optimal parameterization of the hypothesis space, and then optimize a linear predictor from this hypothesis space for the target task. Ando and Zhang's algorithm serves as the basis of structural correspondence learning (SCL), a transfer learning approach that assumes the availability of labeled data only in the source tasks and little or no supervision in the target task (Blitzer et al., 2006). SCL has been applied to natural language problems such as part-of-speech tagging

(Blitzer et al., 2006) and sentiment classification (Blitzer, Dredze, & Pereira, 2007) and operates by inducing a mapping between the feature spaces of the source and target domains. This mapping is produced by using so-called pivot features that behave identically in the source and target tasks; the non-pivot features are mapped across the two domains if they correlate with many of the same pivot features.

3.1.2 Single-task Transfer Learning

In an alternative transfer setting, tasks are presented to the learner one by one, and the goal is to improve learning on the current, target, task by utilizing knowledge acquired in previous learning domains. One of the earliest approaches, the TC Algorithm by Thrun and O'Sullivan (1996), improves target task performance of a nearest-neighbor algorithm by transferring the distance metrics learned on related problems over the same feature space. An interesting aspect of the TC algorithm is that, rather than assuming that the previously-encountered tasks are similar, it autonomously determines task relatedness by using a validation set to estimate how likely it is that a distance metric optimized for a previous task improves performance on the target task. Bonilla et al. (2006) propose a method for transfer learning for estimation of distribution algorithms (EDA) in which a solution to an optimization problem is found by progressively developing a distribution over solutions that estimates the likelihood that a particular solution is optimal. In their work, transfer is achieved by initializing the EDA algorithm with the solution distribution of previously-solved problems. This is done by either combining the predictive distributions from all previous problems or from the k most similar ones, which are found using a k-nearest-neighbor algorithm. Raina, Ng, and Koller (2006) use several related source tasks to construct the covariance matrix for a Gaussian prior in a text classification task.

Transfer learning approaches have also been developed for reinforcement learning (e.g., Taylor, 2008). For example, Taylor, Stone, and Liu (2005) use the value function learned in a source task to initialize reinforcement learning in the target task. Value function transfer is also used by Banerjee and Stone (2007) to transfer knowledge across game-playing domains by using state features extracted from look-ahead game trees. Torrey, Walker, Shavlik, and Maclin (2005) propose extracting advice from the value function learned in the source task, which is then provided to a reinforcement learner in the target task. Taylor, Whiteson, and Stone (2007) propose using policies learned in the source task to direct reinforcement learning in the target task in a more promising direction.

The reinforcement learning community has also studied transfer of relational models (e.g., Torrey, 2009). In particular, in a series of works, Torrey et al. have used relational representations to improve the performance of a reinforcement learning agent in a variety of ways: by using relational macros to learn general descriptions of successful strategies in the source task (2007); by transferring a Q-function represented using an MLN (2008); and, most recently, to transfer a policy represented as an MLN (2009). In related work, Croonenborghs et al. (2007) have introduced an algorithm that learns relational options to aid relational reinforcement learning. Guestrin, Koller, Gearhart, and Kanodia (2003) use relational representations as a vehicle for transfer in planning domains.

The work most relevant to this thesis concerns transfer across relational domains. Davis and Domingos (2008, 2009) use second-order Markov logic[1] to develop DTM, an approach that performs transfer across relational domains that are potentially very different on the surface by learning second-order "clique templates" that capture general regularities, useful across a variety of domains. Second-order MLNs are crucial in this respect because they provide a representation independent of the one used in the source task. Another important characteristic of DTM is that it uses a special learning procedure in the source task in order to increase the likelihood that the acquired knowledge is useful across domains; this is in contrast to approaches like the ones introduced in this chapter that focus on how to make the most out of a pre-existing model, learned to maximize performance specifically on the source task.

3.1.3 Transfer as Mapping

In some cases, successful transfer requires the representation of the source domain to be mapped to that of the target domain. One possibility is for the human designer to provide a hand-constructed mapping (e.g., Taylor et al., 2005). A more widely applicable approach, however, is one that automatically induces a useful mapping (e.g., Blitzer et al., 2006; Y. Liu & Stone, 2006; Taylor, Kuhlmann, & Stone, 2008). Closest to the research we present in this chapter is the structure-mapping engine (SME) (Falkenhainer, Forbus, & Gentner, 1989; Forbus & Oblinger, 1990).

[1]In second-order models, one considers variables over the predicates in the domain, not just over the constants.

SME discovers global one-to-one mappings between the relations and entities in two domains by combining consistent local mappings. Local mappings are formed by matching structural knowledge between the two domains and require at least some information on the structure of the target domain, i.e., the dependencies among its relations. Mappings are evaluated based on a syntactic, structural criterion, called *systematicity*, which does not consider the accuracy of the resulting inferences in the target data.

Y. Liu and Stone (2006) have adapted SME to perform transfer across reinforcement learning tasks whose dynamics are described as qualitative dynamic Bayesian networks (QDBNs).[2] They test their method, which works by automatically mapping the structure of the source and target-task QDBNs, on transfer across simulated robotic soccer domains.

3.1.4 Transfer as Revision

Transfer learning can also be approached as a revision task, in which the source knowledge is viewed as a partially correct model that needs to be refined. Revision algorithms have been developed for a variety of learning models. One such algorithm, FORTE, was described in Section 2.2.4. FORTE has been recently extended by Duboc, Paes, and Zaverucha (2008) to allow for large speed-ups while maintaining the accuracy of the revised theories. Paes et al. (2005) extended FORTE to allow it to handle Bayesian logic programs (Kersting & De Raedt, 2001). These FORTE-based algorithms first diagnose the provided model and then focus the search

[2] A QDBN is a dynamic Bayesian network that can have links of different types.

for revisions on the potentially faulty regions. An analogous approach is used by Ramachandran and Mooney (1998) for revision of Bayesian networks where the source networks are instrumented with leak nodes that are then used as indicators for errors. None of these previous works, however, were applied to transfer learning.

3.2 RTAMAR: When Target-Domain Data is Sufficient

The problem of transferring the structure of an MLN from a source to a target domain can be viewed as consisting of two parts. First, in order to translate the source structure to the target domain, a correspondence between the predicates of the source domain and those of the target domain needs to be established. Second, once the source structure has been translated, it needs to be revised in order to adapt it to the target domain.

This section focuses on solving the second problem and describes our algorithm for revising the structure of the source MLN when an adequate amount of data from the target domain is available. The algorithm assumes that the predicates in the source structure have been mapped to the target domain. This is a safe assumption because this mapping capability was developed by Tuyen Huynh as part of TAMAR, a complete transfer system (Mihalkova, Huynh, & Mooney, 2007). We will call the mapping portion of TAMAR, MTAMAR. MTAMAR uses the concept of a *type-consistent* mapping. A mapping of a source clause to the target domain implies a correspondence from the source predicates in the clause to a subset of the target predicates. Such a correspondence between a source predicate and a target predicate implicitly defines a mapping between the types of the arguments of the two predicates. A mapping is *type-consistent* if, within a clause, a type in the source domain is mapped to at most one type in the target domain. MTAMAR maps each source clause independently of the others by evaluating all possible type-consistent mappings with the WPLL score (Kok & Domingos, 2005) (described on page 31), computed on the target data. The mapping that achieves the highest score is output.

Example 3.2.1. To illustrate MTAMAR, we consider transfer from an academic domain, which contains information about the students and professors in a department, their publications, advising relationships, teaching activities, etc., to a domain about the movie business, such as the one we considered in Example 2.1.1. These two domains use different representations, i.e. distinct sets of predicates, but because they both concern human interactions, we expect there to be significant similarities between them that would make transfer learning beneficial. Figure 3.1 (Mihalkova et al., 2007) shows an instance of such transfer, in which a single clause is being transferred from the source domain. The source clause states that if professor A and student B are authors of the same publication, then A is B's advisor. MTAMAR maps this source clause to the target movie domain, using the best mapping it found, shown in the figure. The resulting mapped clause states that if director A and actor B appeared in the credits of the same movie, then B worked for A.

3.2.1 Revision of MLN Structure for Transfer

Once the source clauses have been mapped to the target domain, they may need to be further revised. This task is carried out by RTAMAR, the revision part of

Source clause:

Publication(T, A) ∧ Publication(T, B) ∧ Professor(A) ∧ Student(B) ∧
¬SamePerson(A, B) ⇒ AdvisedBy(B, A)

Best mapping:

Publication(title,person)	→ Credits(movie,person)
Professor(person)	→ Director(person)
Student(person)	→ Actor(person)
SamePerson(person,person)	→ SamePerson(person,person)
AdvisedBy(person,person)	→ WorkedFor(person,person)

Clause mapped to target domain:

Credits(T, A) ∧ Credits(T, B) ∧ Director(A) ∧ Actor(B) ∧
¬SamePerson(A, B) ⇒ WorkedFor(B, A)

Figure 3.1: An example output of the predicate mapping algorithm

TAMAR. The skeleton of RTAMAR has three steps and is similar to that of FORTE (Richards & Mooney, 1995), which revises first-order theories.

1. **Self-Diagnosis:** The purpose of this step is to focus the search for revisions only on the inaccurate parts of the MLN. The algorithm inspects the source MLN and determines for each clause whether it should be shortened, lengthened, or left as is. For each clause C, this is done by considering every possible implication rewrite of C in which one of the literals is placed on the right-hand side of the implication and is treated as the conclusion and the remaining literals serve as the antecedents. The conclusion of a clause is drawn only if the antecedents are satisfied and the clause “fires.” Thus, if a clause makes the wrong conclusion, it is considered for lengthening because the addition of more literals, or conditions, to the antecedents will make them

harder to satisfy, thus preventing the clause from firing. On the other hand, there may be clauses that fail to draw the correct conclusion because there are too many conditions in the antecedents that prevent them from firing. In this case, we consider shortening the clause.

2. **Structure Update:** Clauses marked as too long are shortened, while those marked as too short are lengthened.

3. **New Clause Discovery:** New clauses are found in the target domain by relational pathfinding (RPF) (Richards & Mooney, 1992).

We next describe each step in more detail.

3.2.1.1 Self-Diagnosis

A natural approach to self-diagnosis is to use the transferred MLN to make inferences in the target domain and observe where its clauses fail. This suggests that the structure can be diagnosed by performing Gibbs sampling over it. Specifically, this is done as follows. Each predicate in the target domain is examined in turn. The current predicate under examination is denoted as P^*. Self-diagnosis performs Gibbs sampling with P^* serving as a query predicate with the values of its gliterals set to unknown, while the gliterals of all other predicates provide evidence. In each round of sampling, in addition to re-sampling a value for gliteral X of P^*, the algorithm considers the set of all ground clauses $\mathcal{G}_X$ in which X participates.

Each ground clause $C \in \mathcal{G}_X$ can be placed in one of four bins with respect to X and the current truth assignments to the rest of the gliterals. These bins consider

Director(coppola) Actor(brando)
Credits(godFather, brando) Credits(godFather, coppola)
Credits(rainMaker, coppola) WorkedFor(brando, coppola)

Figure 3.2: Example relational database

all possible cases of the premises being satisfied and the conclusion being correct. We label a clause as Relevant if the premises are satisfied and Irrelevant otherwise. For positively weighted clauses, we mark a relevant clause as Good if and only if its conclusion is correct, and we mark an irrelevant clause as Good if and only if the conclusion is incorrect. The Good/Bad labels are flipped for clauses with negative weights. The four bins are defined by all possible ways of marking a clause as Relevant/Irrelevant and Good/Bad.

Let v be the actual truth value of X. This value is known from the data, even though for the purposes of sampling we have set it to unknown. As an illustration, we will use some groundings of the clauses in Figure 3.3 with respect to the data in Figure 3.2 (copied from page 18 for convenience) listing the current truth assignments to the gliterals (the ones present are true; the rest are false). Figure 3.3 also lists rewrites of the clauses in implication form where the implication has the target predicate as a conclusion. This will be helpful in the exposition of the algorithm. Let $X = \texttt{Actor}(\texttt{brando})$ with $v = \texttt{true}$. The following descriptions assume positive weights. The negative weight cases are symmetric.

Relevant; Good: This bin contains clauses in which the premises are satisfied and the conclusion drawn is correct. For example, if C is a grounding of the

Clausal form	Implication form wrt target predicate $Actor$
Director(A)$\vee$Actor(A)	$\neg$Director(A) $\Rightarrow$ Actor(A)
Credits(M, A)$\vee\neg$Actor(A)	$\neg$Credits(M, A) $\Rightarrow$ $\neg$Actor(A)
$\neg$WorkedFor(B, A)$\vee\neg$Actor(A)	WorkedFor(B, A) $\Rightarrow$ $\neg$Actor(A)
Actor(A)$\vee\neg$Credits(M,A)$\vee\neg$WorkedFor(A, B)	Credits(M, A) $\wedge$ WorkedFor(A, B) $\Rightarrow$ Actor(A)

Figure 3.3: Clauses in example MLN for diagnosis

first clause in Figure 3.3 with the constant `brando`, i.e. in implication form, $\neg$Director(brando) $\Rightarrow$ Actor(brando), it falls in this bin. We can alternatively describe clauses in this bin as ones which are satisfied only if X has truth value v, the value it has in the data.

Relevant; Bad: The clauses in this bin are those whose premises are satisfied but the conclusion drawn is incorrect. One such clause is $\neg$Credits(rainMaker, brando) $\Rightarrow$ $\neg$Actor(brando). Considering the clausal form, Credits(rainMaker, brando) $\vee\neg$Actor(brando), we see that this bin contains clauses that are only satisfied if X has value $\neg v$, the negation of its correct value in the data.

Irrelevant; Good: This bin contains clauses whose premises are not satisfied, and therefore the clauses do not "fire," but if they were to fire, the conclusion drawn would be incorrect. One such clause is WorkedFor(coppola, brando) $\Rightarrow$ $\neg$Actor(brando). In clausal form this formula is $\neg$WorkedFor(coppola, brando) $\vee\neg$Actor(brando). Thus a more mechanical way of describing the clauses in this bin is that they are satisfied regardless of the value of X in the data; however, the literal corresponding to X in C is true only if X has value $\neg v$.

Irrelevant; Bad: The clauses in this bin are those whose premises are not satisfied, but if the clauses were to fire, the conclusion would be correct. One such clause is Credits(rainMaker, brando) $\wedge$ WorkedFor(brando, coppola) $\Rightarrow$ Actor(brando). If we consider the clausal form, Actor(brando) $\vee\neg$Credits (rainMaker, brando) $\vee$ $\neg$WorkedFor(brando, coppola), we can alternatively describe the clauses in this bin as ones that are satisfied regardless of the value of X and in which the literal corresponding to X in C is true only if X has value v.

Note that, although our examples only use clauses that contain a single literal of P^*, the algorithm handles clauses with multiple P^* literals by setting the ones appearing in the premises to their truth values from the current iteration of Gibbs sampling.

This taxonomy is motivated by Equation 2.4, reprinted here for convenience:

$$P(X = x|\text{MB}_X = \mathbf{m}) = \frac{e^{S_X(x,\mathbf{m})}}{e^{S_X(0,\mathbf{m})} + e^{S_X(1,\mathbf{m})}} \tag{3.1}$$

The probability of $\mathbf{X} = \mathbf{x}$ is increased only by clauses in the **[Relevant; Good]** bin and is decreased by clauses in the **[Relevant; Bad]** bin. Clauses in the other two bins do not have an effect on this equation because their contribution to the numerator and denominator cancels out. To see how this happens, consider a clause $g_{irr} \in \mathcal{G}_X$ from the set of ground clauses in which X participates, such that g_{irr} is satisfied regardless of the truth value of X. The quantity $S_X(x, \mathbf{m})$ from Equation 3.1 can be rewritten as follows:

$$\begin{aligned}
S_X(x, \mathbf{m}) &= \sum_{g_i \in \mathcal{G}_X} w_i g_i(X = x, \mathrm{MB}_X = \mathbf{m}) && (3.2)\\
&= \sum_{g_i \in \mathcal{G}_X, i \neq irr} w_i g_i(X = x, \mathrm{MB}_X = \mathbf{m}) \\
&\qquad + w_{irr} g_{irr}(X = x, \mathrm{MB}_X = \mathbf{m}) && (3.3)\\
&= \sum_{g_i \in \mathcal{G}_X, i \neq irr} w_i g_i(X = x, \mathrm{MB}_X = \mathbf{m}) + w_{irr} && (3.4)\\
&= S^*_X(x, \mathbf{m}) + w_{irr} && (3.5)
\end{aligned}$$

The next-to-last line follows because g_{irr} was picked such that the value of $g_{irr}(X = x, \mathrm{MB}_X = \mathbf{m})$ is 1 regardless of x. Using this derivation, we can rewrite Equation 2.4 as follows:

$$\begin{aligned}
P(X = x | \mathrm{MB}_X = \mathbf{m}) &= \frac{e^{S_X(x,\mathbf{m})}}{e^{S_X(0,\mathbf{m})} + e^{S_X(1,\mathbf{m})}} && (3.6)\\
&= \frac{e^{S^*_X(x,\mathbf{m})+w_{irr}}}{e^{S^*_X(0,\mathbf{m})+w_{irr}} + e^{S^*_X(1,\mathbf{m})+w_{irr}}} && (3.7)\\
&= \frac{e^{w_{irr}} e^{S^*_X(x,\mathbf{m})}}{e^{w_{irr}} \left(e^{S^*_X(0,\mathbf{m})} + e^{S^*_X(1,\mathbf{m})} \right)} && (3.8)
\end{aligned}$$

As can be seen in line 3.8, the contribution of g_{irr}, $e^{w_{irr}}$, can be canceled from the numerator and denominator.

If some of the literals other than X in an **[Irrelevant; Bad]** clause, are deleted so that X's value becomes crucial, it will be moved to the **[Relevant; Good]** bin. Similarly, if we add some literals to a **[Relevant; Bad]** clause so that it starts to hold regardless of the value of X, it will enter the **[Irrelevant; Good]** bin and will no longer decrease the probability of X having its correct value.

As the value of a gliteral is re-sampled in each iteration of Gibbs sampling, for each clause in which the gliteral participates, we count the number of times it falls into each of the four bins. Finally, if a clause was placed in the **[Relevant; Bad]** bin more than p percent of the time, it is marked for lengthening and if it fell in the **[Irrelevant; Bad]** bin more than p percent of the time, it is marked for shortening. We anticipated that in the highly sparse relational domains in which we tested, clauses would fall mostly in the **[Irrelevant; Good]** bin. To prevent this bin from swamping the other ones, we set p to the low value of 10%. This value was set during earlier experiments on artificial data (Mihalkova & Mooney, 2006) and was not tuned to the data used for the experiments presented here. In the future, it would be interesting to consider ways in which such parameters can be set automatically. The process described above is repeated for each predicate, P^*, in the target domain.

3.2.1.2 Structure Updates

Once the set of clauses to revise is determined, the actual updates are performed using beam search. Beam search proceeds in iterations. In each iteration, all possible literal additions or deletions are performed to the set of current candidates, the n best-performing are kept, and a new iteration begins. Unlike Kok and Domingos (2005), however, we do not consider all possible additions and deletions of a literal to each clause. Rather, we only try removing literals from the clauses marked for shortening and we try literal additions only to the clauses marked for lengthening. The candidates are scored using WPLL. Thus, the search space is constrained first by limiting the number of clauses considered for updates, and second, by restricting the kind of update performed on each clause.

3.2.1.3 New Clause Discovery

The revision procedure can update clauses transferred from the source domain but cannot discover new clauses that capture relationships specific only to the target domain. To address this problem, we used RPF (Richards & Mooney, 1992) (described in Section 2.2.3) to search for new clauses in the target domain. The clauses found by RPF were evaluated using WPLL, and the ones that improved the overall score were added to the MLN. RPF and the previous structure updates step operate independently of each other; in particular, the clauses discovered by RPF are not diagnosed nor revised. However, we found that better results are obtained if the clauses discovered by RPF are added to the MLN before carrying out the revisions. This can be explained as follows. The revision step fills the resulting structure with clauses that together achieve a very good WPLL on the training data. If we perform RPF after this, even though it finds clauses that are very reasonable and would perform quite well, the MLN already has other clauses that interfere. In this way, the good clauses discovered by RPF sometimes end up not being added. On the other hand, if we first add the RPF clauses to the MLN, they give an initial boost in WPLL and also constrain the beam search, causing it to finish faster because it has less to improve.

3.2.2 Experiments

In this section we present an experimental evaluation of TAMAR. First, we describe our methodology, which will also be used for the experiments in Chapter 4. We then discuss the results specific to TAMAR.

Experimental Methodology: We used three relational domains—IMDB, UW-CSE, and WebKB. Each data set is broken into *mega-examples*, where each mega-example contains a connected group of facts. Individual mega-examples are independent of each other. By arranging multi-relational data into mega-examples, we are able to carry out principled cross-validation experiments, where, because mega-examples are independent of one another, we can provide some as training data and test on the rest. This is preferable to breaking up mega-examples, because in the latter case, it is not clear how to break up the relations in the data so that there is sufficient information for training and the test data is not contaminated.

The IMDB database is organized as five mega-examples, each of which contains information about four movies, their directors, and the first-billed actors who appear in them. Each director is ascribed genres based on the genres of the movies he or she directed. The Gender predicate is only used to state the genders of actors. The complete list of predicates in this domain is given in Figure 3.4 (a). This data set[3] is dramatically smaller than the data available from the International Movie Database (`www.imdb.com`). The reason for this is that originally the data set was intended to be used as a target domain in which data is limited.

[3]Available from `http://www.cs.utexas.edu/~ml/mlns` under "Data sets."

The UW-CSE database was compiled by Richardson and Domingos (2006).[4] It lists facts about people in an academic department (i.e. `Student`, `Professor`) and their relationships (i.e. `AdvisedBy`). The complete list of predicates is given in Figure 3.4 (b). The database is divided into mega-examples based on five areas of computer science.

The WebKB database contains information about human relationships from the "University Computer Science Department" data set, compiled by Craven et al. (1998). The original data set contains Web pages from four universities labeled according to the entity they describe (e.g. student, course), as well as the words that occur in these pages. Our version of WebKB[5] contains the predicates listed in Figure 3.4 (c). The textual information is ignored. This data contains four mega-examples, each of which describes one university. To extract the truth values for these predicates, we used the files from the original data set that list the student, faculty, instructors-of, and members-of-project relationships. We treated each Web address in these files as an entity in the domain and used the label of the corresponding page to determine the gliteral truth values. Table 3.1 provides additional information about the domains.

To evaluate a given MLN, one needs to perform inference over it, providing some of the gliterals in the test mega-example as evidence and testing the predictions of the remaining ones. We followed the testing scheme employed by Kok and Domingos (2005) and tested for the gliterals of each of the predicates of

[4]Available at `http://alchemy.cs.washington.edu/` under "Datasets."
[5]Available at `http://www.cs.utexas.edu/~ml/mlns/` under "Data sets."

(a) Predicates in IMDB
Director(person)
Actor(person)
Movie(title, person)
Gender(person, gend)
WorkedUnder(person, person)
Genre(person, genr)
SamePerson(person, person)
SameMovie(title, title)
SameGenre(genr, genr)
SameGender(gend, gend)

(b) Predicates in UW-CSE
TaughtBy(course, person, semester)
CourseLevel(course, level)
Position(person, pos)
AdvisedBy(person, person)
ProjectMember(project, person)
Phase(person, phas)
TempAdvisedBy(person, person)
YearsInProgram(person, year)
TA(course, person, semester)
Student(person)
Professor(person)
SamePerson(person, person)
SameCourse(course, course)
SameProject(project, project)
Publication(title, person)

(c) Predicates in WebKB
Student(person)
SamePerson(person, person)
Faculty(person)
Project(projname, person)
CourseTA(coursename, person)
CourseProf(coursename, person)

Figure 3.4: Predicates in each of our domains. The argument types for each predicate are listed in the parentheses.

the domain in turn, providing the rest as evidence, and averaging over the results. However, for inference we used the MC-SAT algorithm that has been demonstrated to give more accurate results (Poon & Domingos, 2006). The inference procedure outputs the probability that each of the query gliterals is true. To summarize these results, we used two standard evaluation metrics common in the SRL community that were also employed by Kok and Domingos (2005): the area under the precision-recall curve (AUC) and the conditional log-likelihood (CLL). To compute the AUC, first a precision-recall curve is generated. This is done by varying a

Data Set	Number of Consts	Number of Types	Number of Preds	Number of True Gliterals	Total Number of Gliterals
IMDB	316	4	10	1,540	32,615
UW-CSE	1,323	9	15	2,673	678,899
WebKB	1,700	3	6	2,065	688,193

Table 3.1: Details about the domains.

probability threshold whose value determines which propositions are labeled positive and which negative; i.e. the ones whose probability of being true is greater than the threshold are positive and the rest are negative. The precision and recall are computed as follows:

$$\text{Precision} = \frac{\text{Number of propositions correctly labeled as positive}}{\text{Number of all propositions labeled as positive}}$$

$$\text{Recall} = \frac{\text{Number of propositions correctly labeled as positive}}{\text{Total number of positive propositions in the data}}$$

A curve is produced by plotting a point for the precision and recall obtained at a set of threshold values. The AUC is the area under this curve. The AUC is useful because it demonstrates how well the algorithm predicts the few positives in the data and is not affected by the large number of true negatives typically present in relational data sets (the reader is encouraged to compare the number of true gliterals to the total number of gliterals in Table 3.1).

The CLL is computed by taking the log of the probability predicted by the model that a gliteral has its correct truth value in the data, and averaging over the query gliterals. The CLL complements the AUC because it determines the quality of the probability predictions output by the algorithm.

Learning curves for each performance measure were generated using a leave-1-mega-example-out approach, averaging over k different runs, where k is the number of mega-examples in the domain. In each run, we reserved a different mega-example for testing and trained on the remaining $k - 1$, which were provided one by one. All systems observed the same sequence of mega-examples. The error bars on the curves are formed by averaging the standard error over the predictions for the groundings of each predicate and over the learning runs. Error bars are drawn on all curves but in some cases they are tiny.

We also present results on the training times needed by the learners, and the number of clauses they considered in their search. Timing runs within the same transfer experiment were conducted on the same dedicated machine.

Systems Compared: We compared the performance of the following systems. KD run from scratch (ScrKD) in the target domain; KD used to revise a source structure translated into the target domain using MTAMAR (TrKD); and the complete transfer system using MTAMAR and RTAMAR (TAMAR).

We used the implementation of KD provided as part of the Alchemy software package (Kok et al., 2005) and implemented our new algorithms as part of the same package. We kept the default parameter settings of Alchemy except that we set the parameter penalizing long clauses to 0.01, the one specifying the maximum number of variables per clause to 5, and the minWeight parameter to 0.1 in IMDB and WebKB and to 1 in UW-CSE, the value used in (Kok & Domingos, 2005). All three learners used the same parameter settings.

We considered the following transfer scenarios: WebKB $\rightarrow$ IMDB, UW-CSE $\rightarrow$ IMDB, WebKB $\rightarrow$ UW-CSE, IMDB $\rightarrow$ UW-CSE, where the notation Domain 1 $\rightarrow$ Domain 2 means transfer from Domain 1 to Domain 2. We did not consider transfer to WebKB because the small number of predicates and large number of constants in each mega-example, which represents an entire university, made it too easy to learn from scratch in this domain. WebKB is therefore a good source domain but uninteresting as a target domain. Source MLNs were learned by ScrKD. We also consider the scenario where the hand-built knowledge base provided with the UW-CSE data is used as a source MLN (UW-KB $\rightarrow$ IMDB). This knowledge base was written by human volunteers who were instructed to write general facts about academia in first-order logic (Richardson, 2004). In this interesting twist on traditional theory refinement, the provided theory needs to be mapped to the target domain, as well as revised.

Results: The full learning curves are presented in Appendix 1. Here we summarize them using two statistics: the transfer ratio (TR) (Cohen, Chang, & Morrison, 2007), and the percent improvement from 1 mega-example (PI). TR is the ratio between the area under the learning curve of the transfer learner (TAMAR or TrKD) and the area under the learning curve of the learner from scratch (ScrKD). Thus, TR gives an overall idea of the improvement achieved by transfer over learning from scratch. TR > 1 signifies improvement over learning from scratch in the target domain. PI gives the percent by which transfer improves accuracy over learning from scratch after observing a single mega-example in the target domain. It is useful be-

	TR		PI	
Experiment	TrKD	TAMAR	TrKD	TAMAR
WebKB → IMDB	1.51	1.55	50.54	53.90
UW-CSE → IMDB	1.42	1.66	32.78	52.87
UW-KB → IMDB	1.61	1.52	40.06	45.74
WebKB → UW-CSE	1.84	1.78	47.04	37.43
IMDB → UW-CSE	0.96	1.01	-1.70	-2.40
Average	1.47	1.50	33.74	37.51

Table 3.2: Transfer ratio (TR) and percent improvement from 1 mega-example (PI) on **AUC** over ScrKD.

cause in transfer-learning settings data for the target domain is frequently limited.

In terms of AUC (Table 3.2), both transfer systems improve over ScrKD in all but one experiment. Neither transfer learner consistently outperforms the other on this metric, but on average over the five experiments, TAMAR performs slightly better. We note that in transfer to UW-CSE, TAMAR's PI is smaller than that of TrKD, even though their TRs are roughly the same. We conjecture that this happens because the mega-examples in UW-CSE are not identically distributed. Each mega-example in this domain represents one area of computer science, and the types and amounts of interaction among the entities vary across areas. As a result, when TAMAR uses just one of these mega-example to self-diagnose the source structure, it may be misled by the peculiarities of that mega-example, causing it to mis-assign source clauses to bins. This is corroborated by the fact that the performance of TAMAR and TrKD becomes roughly the same when more mega-examples are provided, as indicated by the roughly equal TRs and as can also be seen from the full learning curves in Appendix 1.

	TR		PI	
Experiment	TrKD	TAMAR	TrKD	TAMAR
WebKB → IMDB	1.41	1.46	51.97	67.19
UW-CSE → IMDB	1.33	1.56	49.55	69.28
UW-KB → IMDB	1.21	1.44	30.66	58.62
WebKB → UW-CSE	1.17	1.36	19.48	32.69
IMDB → UW-CSE	1.62	1.67	34.69	54.02
Average	1.35	1.50	37.27	56.36

Table 3.3: Transfer ratio (TR) and percent improvement from 1 mega-example (PI) on **CLL** over ScrKD.

Table 3.3 shows that transfer learning always improves over learning from scratch in terms of CLL, and TAMAR's performance is better than TrKD's in all cases. In the last experiment, IMDB → UW-CSE, we observe that transfer improves over learning from scratch in terms of CLL but is worse in terms of AUC. This demonstrates that AUC and CLL complement each other. We believe this slightly worse performance of the transfer systems is probably due to random variation.

Moreover, as can be seen from Table 3.4, TAMAR trains faster than TrKD, and both transfer systems are faster than ScrKD. TAMAR also considers fewer candidate clauses during its beam search, as can be seen in Table 3.5. According to a t-test performed for each point on each of the learning curves, at the 95% level with sample size 5 per point, these differences were significant in 15 out of 20 cases for speed and 18 out of 20 for number of candidates. In some cases TrKD considers more candidates than ScrKD but takes less time to train. This can happen if TrKD considers more candidates earlier in the learning curves when each candidate is evaluated faster on less data.

Experiment	ScrKD	TrKD	TAMAR	TAMAR speed-up over TrKD
WebKB→IMDB	62.23	32.20	11.98	2.69
UW-CSE→IMDB	62.23	38.09	15.21	2.50
UW-KB→IMDB	62.23	40.67	6.57	6.19
WebKB→UW-CSE	1127.48	720.02	13.70	52.56
IMDB→UW-CSE	1127.48	440.21	34.57	12.73
Average	488.33	254.24	16.41	15.33

Table 3.4: Average (over all learning curve points) total training time in minutes.

Experiment	ScrKD	TrKD	TAMAR
WebKB→IMDB	7,558	10,673	1,946
UW-CSE→IMDB	7,558	14,163	1,976
UW-KB→IMDB	7,558	15,118	1,613
WebKB→UW-CSE	32,096	32,815	827
IMDB→UW-CSE	32,096	7,924	978
Average	17,373.2	16,138.6	1,468.0

Table 3.5: Average (over all learning curve points) number of candidate clauses evaluated.

The complete learning curves are given in Appendix 1. Here we present the most interesting among them. Figure 3.5 shows the learning curve in the UW-CSE → IMDB experiment. Here we additionally tested the performance of systems that do not use MTAMAR but are provided with an intuitive hand-constructed mapping that maps Student → Actor, Professor → Director, AdvisedBy/TempAdvisedBy → WorkedFor, Publication → Movie, Phase → Gender, and Position → Genre. The last two mappings are motivated by the observation that Phase in UW-CSE applies only to Student and Gender in IMDB applies only to Actor, and similarly Position and Genre apply only to Professor and Director respectively. The systems using the automatic mapping perform much better because MTAMAR maps each clause

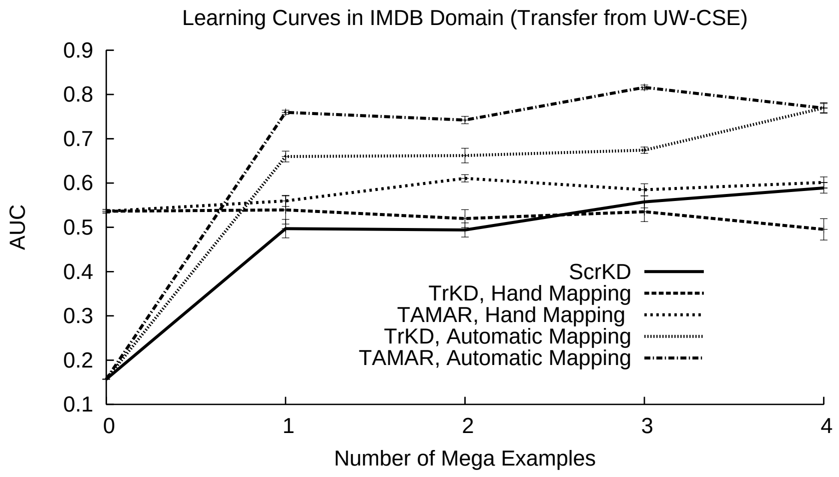


Figure 3.5: Learning curves in UW-CSE → IMDB for AUC. The zeroth points are obtained by testing the MLN provided to the learner at the start.

independently of the rest; i.e., the same source predicate appearing in different clauses may be mapped in different ways. MTAMAR also has the ability to "erase" a predicate from a clause by mapping it to the "empty" predicate in the target domain. This flexibility allows the source knowledge to adapt better to the target domain.

3.3 SR2LR: When Target-Domain Data is Severely Limited

TAMAR assumes that at least one mega-example from the target domain is available. In this section, we study the challenging case of limited target data, in which transfer learning could have the greatest impact. In particular, here we assume minimal target-domain data that consists of just a handful of entities, in the extreme case just a single one. Figure 3.6 contrasts the amount of data assumed by TAMAR to that assumed in this section.

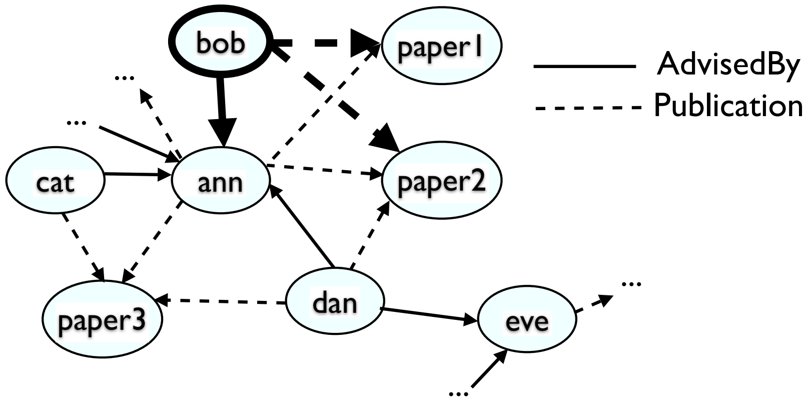


Figure 3.6: Target data assumed by SR2LR vs TAMAR. The nodes in this graph represent the entities in the domain and the edges represent the relations in which these entities participate. TAMAR assumes that the information from the entire graph is provided. SR2LR assumes that just the bold relations are known.

This setting may arise in a variety of situations. For instance, when a new social networking site is launched, data is available on only a few initial registrants. The popularity of the site depends on its ability to make meaningful predictions that would allow it to suggest promising friendships to users. However, the sparsity of available data and the fact that data from other social networking sites is usually proprietary make learning of an effective model from scratch infeasible.

Frequently, two domains differ in their representations, but the underlying regularities that govern the dynamics in each domain are similar. So, when transferring a model learned from an academic data set to a movie business domain, one may discover that students and professors are similar to actors and directors respectively, which makes writing an academic paper analogous to directing or participating in a movie. Likewise, because human interactions bear a certain degree of similarity across settings, the social networking site can learn strong models from data on the professional relations among its employees and map them for the task of interest based on its very limited supply of data from the new site.

3.3.1 The SR2LR Algorithm

When target data is so limited, effective transfer depends on the ability to map the representation of a source model learned in a closely related domain to that of the target task. The main challenge addressed in this section is, therefore, to harness the small amount of data in the target domain in order to find useful mappings between the source and target representations.

We present an efficient algorithm for this task, SR2LR (which stands for Short-Range To Long-Range) (Mihalkova & Mooney, 2009b), that is based on the observation that a good model for the source domain contains two types of clauses—short-range ones that concern the properties of a single entity and long-range ones that relate the properties of several entities. Because possible mappings of the short-range clauses to the target domain can be directly evaluated on the available target data, the key is to use the short-range clauses in order to find mappings between the relations in the two domains, which are then used to translate the long-range clauses, thus boosting the performance of the model in the target domain.

Single-Entity Case: We first describe the algorithm for the extreme *single-entity-centered* setting, in which information about only one entity is available. Then we generalize to more than one entity. More precisely, for now we assume that the data lists all true gliterals concerning a **central** entity e, and only those gliterals. Gliterals that involve e but are not listed are assumed to be false. Gliterals that do not involve e have unknown values. Thus, unlike in other sections, here we are making the closed-world assumption *only* with respect to the central entity. Facts that are not about the central entity are assumed to have unknown, rather than false, truth values.

Mapped clauses that can be directly evaluated given a single-entity-centered example are short-range; the rest are long-range.

Definition 3.3.1. A clause C is **short-range** with respect to an entity of type t iff there exists a variable v that appears in every literal of C and v represents arguments of type t. A clause is **long-range** with respect to E iff it is not short-range.

Example 3.3.1. As an example, suppose we would like to transfer the MLN in Figure 3.7 using the data in Figure 3.8, i.e., transfer from a movie domain to an academic domain. Let us consider one possible type-consistent mapping of the first clause in Figure 3.7, which is given in line 1.1 of Figure 3.9. Note that variable A appears in both literals of this clause. Therefore, the clause is short-range. The truth value of any grounding that uses the substitution $A = bob$ can be directly evaluated from the data. For example, if we ground this clause using the substitution $A = bob, B = ann$, we obtain a ground clause whose literals are all known from our data, thus the clause can be evaluated and hence, it is short-range.

Definition 3.3.2. A ground clause is **verifiable** if it contains only gliterals with known truth values.

Example 3.3.2. Continuing the example, if we use the substitution $A = ann, B = bob$, the resulting grounding cannot be directly evaluated because the truth-value of

1	0.7 : WorkedFor$(A, B) \Rightarrow \neg$Director(A)
2	0.8 : Credits$(M, A) \wedge$ Credits$(M, B) \wedge$ Director$(B) \Rightarrow$ WorkedFor(A, B)

Figure 3.7: Source MLN

Student(bob), Publication(paper1, bob), Publication(paper2, bob) , AdvisedBy(bob, ann)

Figure 3.8: Target domain data centered around `bob.` All listed atoms are true; atoms about `bob` that are not listed are false; the remaining atoms have unknown values.

`Professor`(ann) is unknown. We say that the earlier grounding is **verifiable,** whereas the second one is not. Now consider one possible mapping of the second clause in Figure 3.7, given in line 2.1 of Figure 3.9. This clause concerns relations that go beyond just a single entity because it is about papers written by other people and is therefore long-range.

Algorithm 1 formally describes SR2LR. In line 1, the weight of a mapped clause is set to the weight of the source clause from which it was mapped. Because of limited target data, we do not attempt to re-learn weights or to revise the mapped clauses.[6] In line 3, the short-range mapped clauses are evaluated, as described in Algorithm 2, which checks whether the verifiable groundings of short-range clauses are satisfied in the target data. Clauses that are satisfied at least Θ proportion of the time are accepted; the rest are rejected. This procedure automatically rejects clauses that are not informative.

[6]MTAMAR also directly copies the weights.

1.1	AdvisedBy$(A, B) \Rightarrow \neg$Professor(A)
WorkedFor $\rightarrow$ AdvisedBy, Director $\rightarrow$ Professor	
1.2	AdvisedBy$(A, B) \Rightarrow \neg$Student(A)
WorkedFor $\rightarrow$ AdvisedBy, Director $\rightarrow$ Student	
2.1	Publication$(M, A) \wedge$ Publication$(M, B) \wedge$ Professor$(B) \Rightarrow$ AdvisedBy(A, B)
WorkedFor $\rightarrow$ AdvisedBy, Director $\rightarrow$ Professor, Credits $\rightarrow$ Publication	
2.2	Publication$(M, A) \wedge$ Publication$(M, B) \wedge$ Student$(A) \Rightarrow$ AdvisedBy(A, B)
WorkedFor $\rightarrow$ AdvisedBy, Director $\rightarrow$ Student, Credits $\rightarrow$ Publication	

Figure 3.9: Example mapped clauses. The predicate correspondences used to map each clause are listed under it.

Definition 3.3.3. A short-range clause is **informative** with respect to a single-entity-centered example if it has a verifiable grounding in which at least one gliteral is false.

Intuitively, a clause is uninformative if, in every possible re-writing of the clause as an implication, the premises are never satisfied, and so the clause is always *trivially* true.

Example 3.3.3. For example, consider the clause Student$(A) \vee \neg$AdvisedBy(B, A), which has two verifiable groundings corresponding to the substitutions $A = bob$, $B = ann$, and $A = bob$, $B = bob$. It is not informative because all the literals in its verifiable groundings are true. To develop intuition for the significance of this, consider one of the groundings: Student$(bob) \vee \neg$AdvisedBy(ann, bob). We can re-write it as $\neg$Student$(bob) \Rightarrow \neg$AdvisedBy(ann, bob) or equivalently as AdvisedBy

Algorithm 1 SR2LR algorithm

```
Input: SrcMLN: Source Markov logic network
    T_E: Target data centered on the entity $E$
    $\mathcal{P}$: Set of predicates in the target domain
    $\Theta$: Truth threshold for accepting a short-range clause
Procedure:
 1: Generate TarMap, the set of all possible type-consistent mappings of the clauses
    in SrcMLN. Each mapped clause gets the weight of its corresponding source
    clause.
 2: Split the clauses in TarMap into sets of short-range clauses, $\mathcal{S}$, and long-range
    clauses, $\mathcal{L}$.
 3: $\mathcal{S}'$ = filter-short-range($\mathcal{S}, \Theta$) (Algorithm 2)
 4: Add all clauses from $\mathcal{S}'$ to Result
 5: $\mathcal{L}'$ = filter-long-range($\mathcal{L}, \mathcal{S}'$) (Algorithm 3)
 6: Add all clauses from $\mathcal{L}'$ to Result
 7: Let $\mathcal{A}_C$ be the set of all clauses in Result mapped from source clause $C$ with
    weight $w_C$.
 8: Set the weight of each $a \in \mathcal{A}_C$ to $w_C/|\mathcal{A}_C|$.
```

$(ann, bob) \Rightarrow$ Student(bob). In both cases, the premises of these clauses do not hold, and thus the clauses cannot be used to draw inferences that can be tested. So, judgements about mappings based on such clauses are likely to be misleading.

Once the short-range clauses are evaluated, in line 5 of Algorithm 1, SR2LR evaluates the long-range ones, based on the mappings found to be useful for short-range clauses. A long-range clause is accepted if all source-to-target predicate mappings implied by it either led to accepted short-range clauses (**support by evaluation**) or were never considered by Algorithm 2 (**support by exclusion**). More precisely, let C_S and C_L be short-range and long-range mapped clauses respectively. If the set of source-to-target predicate correspondences implied by C_S is a subset of those implied by C_L, we say that the literals of C_L that appear in C_S are **supported**

Algorithm 2 filter-short-range($\mathcal{S}, \Theta$)

```
1: $\mathcal{S}' = \emptyset$
2: for each $C \in \mathcal{S}$ do
3:     if $C$ is informative and the proportion of verifiable groundings of $C$ that are
       true is $\geq \Theta$ then
4:         Add $C$ to $\mathcal{S}'$
5:     end if
6: end for
7: Return $\mathcal{S}'$
```

Algorithm 3 filter-long-range($\mathcal{L}, \mathcal{S}'$)

```
1: $\mathcal{L}' = \emptyset$
2: for each $LR \in \mathcal{L}$ do
3:     if All literals in $LR$ are supported either by evaluation based on the clauses
       in $\mathcal{S}'$ or by exclusion then
4:         Add $LR$ to $\mathcal{L}'$
5:     end if
6: end for
7: Return $\mathcal{L}'$
```

by evaluation. A correspondence between source predicate P_S and target predicate P_T is **supported by exclusion** with respect to a set of mapped short-range clauses $\mathcal{S}$ if P_S and P_T do not appear in any of the source-to-target predicate correspondences implied by the clauses in $\mathcal{S}$. The goal of support by exclusion is to allow for predicates that do not appear in the short-range clauses to be mapped. Although support by exclusion may seem too risky, i.e., if a pair of completely unrelated source and target predicates are mapped to each other, in our experience the type consistency constraint and the requirement that neither of the predicates was mapped to any other predicate were strong enough to safeguard against this.

Example 3.3.4. We now illustrate Algorithm 1 up to line 7. Figure 3.9 lists some mappings of the clauses in Figure 3.7, along with the source-to-target predicate correspondences implied by them. Clauses 1.1 and 1.2 are (informative) short-range, and 2.1 and 2.2 are long-range. Let $\Theta = 1$. All verifiable groundings of clause 1.1 are satisfied by the target data (given in Figure 3.8). Thus, this clause is accepted and the predicate correspondences found by it are useful. Clause 1.2 is rejected because not all of its verifiable groundings are satisfied by the target data. Thus $\mathbb{S}'$ contains only clause 1.1. Moving on to the long-range clauses, we see that predicates `AdvisedBy` and `Professor` in clause 2.1 are supported by clause 1.1; `Publication` is supported by exclusion, so clause 2.1 is accepted. Clause 2.2 is not accepted because there is no support for `Student(B)`.

Finally, in lines 7-8 of Algorithm 1 the weight of each mapped clause M_C is divided by the number of mapped clauses that originated from the same source clause as M_C in order to ensure that none of the source clauses dominates the resulting model. In preliminary experiments this led to slightly better performance.

More Than One Entity: The generalization to more than one entity is easy. The only difference is that now we have *a set* of single-entity-centered training examples, and Algorithm 2 checks the validity of each short-range clause on each of the examples, accepting a clause if it holds more than Θ proportion of the time over all examples. As more entities become known, some of the long-range clauses become directly verifiable. However, in preliminary experiments, we found that directly evaluating long-range clauses in this way does not significantly help performance, i.e., additional entities lead to improved accuracy mostly because they allow for more reliable evaluation of the short-range clauses.

Choice of Representation: The only characteristic of MLNs crucial to SR2LR is that MLNs use first-order clauses that are interpreted in the standard way for first-order logic, i.e. by evaluating their truth values. This is crucial because SR2LR also interprets the clauses in the traditional way. SR2LR would therefore be applicable to any relational model that is based on a traditional interpretation of first-order logic, such as purely logical representations that perform logical inference, stochastic logic programs (SLPs) (Muggleton, 1996), and MACCENT (Dehaspe, 1997). In SLPs, knowledge is encoded as a set of Horn clauses with attached probabilities. The probability that a particular ground atom is true is calculated by summing the probabilities of all paths in an SLD-tree[7] (De Raedt, 2008) that lead to a successful refutation, where the probability of a path is the product of the probabilities of all clauses that were used in this path. SR2LR could also be applied to transferring knowledge learned by the MACCENT system (Dehaspe, 1997), which is similar to MLNs in that it uses first-order clauses to define a maximum entropy distribution but, unlike MLNs, works only on independent examples and is used to model a conditional distribution. SR2LR would not be applicable to Bayesian logic programs (BLPs)(Kersting & De Raedt, 2001), which do not interpret their clauses in the standard way. Rather, each clause in a BLP encodes a dependency of groundings of the head on the corresponding groundings of the body. MLNs have properties which, while not crucial to SR2LR, contribute to its effectiveness. In particular, the ability of MLNs to handle uncertainty allows SR2LR to recover gracefully from an

[7]An SLD tree shows the steps taken in SLD resolution, a type of logical inference that applies to Horn clauses, in order to prove a given logical statement. A path in this tree represents one possible sequence of steps that can be followed to prove the given statement.

occasional incorrect predicate mapping: provided that most of the mapped clauses are useful, the negative effect of a few misleading ones will be mitigated by the fact that, when computing a probability distribution, MLNs consider the contribution of all of the clauses in the model. This is not observed in purely logical representations in which a clause is used in isolation, and some of the clauses may never be used.

3.3.2 Experiments

We first describe methodology followed in the experiments and then discuss the empirical questions we asked and the results we obtained.

Methodology: We compared SR2LR to MTAMAR and other baselines in the three benchmark relational domains on social interactions that we used to evaluate TAMAR: IMDB, UW-CSE, and WebKB. The IMDB and UW-CSE domains are very similar in terms of the regularities between the relations in them, but the actual representations they use differ. For example, in IMDB an actor and a director are usually in a `WorkedFor` relationship if they appear in the credits of the same movie. Analogously, in UW-CSE a student and a professor are typically in an `AdvisedBy` relationship if they appear in the author list of the same publication. Thus, an algorithm capable of discovering effective mappings from the predicates of one domain to those of the other, would be able to achieve good accuracy via transfer. This example also demonstrates why data centered around a single entity, or a handful of isolated entities, cannot support effective learning from scratch: one of the most useful clauses for predicting `AdvisedBy` involves knowledge about the publications of two *connected* entities, i.e., the advisor and the advisee. Although UW-CSE may seem more closely related to WebKB than to IMDB, in fact, WebKB does not have a predicate analogous to `advisedBy`, which renders it much less useful for transfer. Nevertheless, we include experimental results on transfer from and to WebKB in order observe how the degree of relatedness between the source and target domains affects the quality of transfer. We note that although some of the predicates occur in more than one domain under the same name, the systems do not use the actual predicate names.

Sources were learned with BUSL (Mihalkova & Mooney, 2007), which, as we demonstrate in Chapter 4, gives good performance in the domains we consider.[8] We slightly modified BUSL to encourage it to learn larger models by removing the `minWeight` threshold and by treating the clauses learned for each predicate separately. This leads to models that are less accurate in the source domain but in some cases allow for more effective transfer, as we discovered in preliminary experiments (Mihalkova & Mooney, 2008). We call these models **learned**[9]. Experimental results of transferring from sources learned with the original BUSL are shown in Section 3.3.2.1. For transfer from UW-CSE, we also used the manually coded knowledge base provided with that data set. We call it **manual**.

As before, we report the results in terms of the area under the precision-recall curve (AUC) and the conditional log-likelihood (CLL). We report CLL for

[8]These sources were not used for the experiments with TAMAR because BUSL had not yet been developed at that time.

[9]Source MLNs are available from `http://www.cs.utexas.edu/users/ml/mlns/` under SR2LR.

completeness; however, because we are unable to tune the weights of the MLN on the limited target data, the CLL may be misleading. This can happen when the predicted probabilities are correctly ordered, i.e., true ground atoms have higher probability than false ones (thus giving a high AUC), but are not close to 0 or 1 (thus giving a low CLL). At the same time, because of the large number of true negatives, the CLL can be boosted by predicting near 0 for every ground atom; so a model that predicts very low probabilities has a relatively high CLL even when these probabilities are incorrectly ordered.

We implemented SR2LR and the baselines as part of the Alchemy system (Kok et al., 2005). Θ in Algorithm 1 was set to 1. Inference during testing was performed on the mega-examples other than the one supplying training data, iterating over the available test examples. Within the same experiment, all systems used the same sequence of training and testing examples. The performance of a given predicate was evaluated by inferring probabilities for all of its groundings, given the truth values of all other predicates in the test mega-example as evidence. While training occurs on limited data, we test on a full mega-example. This is appropriate because the final goal of transfer is to obtain a model that gives effective predictions in the target domain as a whole and not just for an isolated entity. For inference, we used the Alchemy implementation of MC-SAT (Poon & Domingos, 2006) with the default parameter settings. Statistical significance was measured via a paired t-test at the 95% level. As a final note, all systems we compared ran extremely efficiently and found mappings in a few seconds on a standard workstation.

Overall Performance: The first set of experiments evaluates the relative accuracy of SR2LR over all predicates in each domain in the most challenging case when only information about a single entity from the target domain is available. We formed single-entity-centered examples by randomly selecting as the central entity 10% of the entities of type person from each mega-example available in the target domain. This resulted in 29 entities in IMDB, 58 in UW-CSE, and 147 in WebKB. We compared against MTAMAR and a **Scratch** baseline that learns with no transfer as follows.

Scratch Baseline: For every ordered pair of known atoms in the available data, a clause is formed by having the first atom imply the second and variablizing consistently. All clauses obtained in this way are assigned a weight of 1. This baseline generates a set of informative clauses that are true in the given data. If a clause has groundings that are violated by the data, then our construction procedure guarantees that there will be another clause with the same weight of 1, which draws the opposite conclusion so that clauses that are not always true in the data cancel each other in pairs during inference. Thus, this baseline can be viewed as a variation of SR2LR that transfers only the short-range clauses of a source model that contains of all possible clauses of length 2.

Tables 3.6 and 3.7 list the accuracies for every possible target/source pair in terms of AUC and CLL respectively. Statistically significant improvement (degradation) over MTAMAR is indicated by a $\uparrow$ ($\downarrow$), and significant improvement (degradation) over Scratch is indicated by $\nearrow$ ($\swarrow$). In terms of AUC, the more informative measure, transfer between UW-CSE and IMDB is always beneficial over learning

Target	Source	MTAMAR	Scratch	SR2LR
IMDB	UW-CSE-learned	0.327	0.276	0.452 ↑ ↗
IMDB	UW-CSE-manual	0.414	0.276	0.577 ↑ ↗
IMDB	WebKB-learned	0.388	0.276	0.468 ↑ ↗
UW-CSE	IMDB-learned	0.115	0.108	0.188 ↑ ↗
UW-CSE	WebKB-learned	0.199	0.108	0.174 ↓ ↗
WebKB	IMDB-learned	0.164	0.287	0.168 ↑ ↙
WebKB	UW-CSE-learned	0.297	0.287	0.295
WebKB	UW-CSE-manual	0.276	0.287	0.178 ↓ ↙

Table 3.6: Average AUC over all target domain predicates.

Target	Source	MTAMAR	Scratch	SR2LR
IMDB	UW-CSE-learned	-1.692	-4.575	-0.682 ↑ ↗
IMDB	UW-CSE-manual	-0.433	-4.575	-0.502 ↓ ↗
IMDB	WebKB-learned	-0.728	-4.575	-0.872 ↓ ↗
UW-CSE	IMDB-learned	-2.057	-5.708	-0.606 ↑ ↗
UW-CSE	WebKB-learned	-1.191	-5.708	-0.891 ↑ ↗
WebKB	IMDB-learned	-1.731	-3.440	-0.694 ↑ ↗
WebKB	UW-CSE-learned	-1.221	-3.440	-0.643 ↑ ↗
WebKB	UW-CSE-manual	-0.561	-3.440	-0.873 ↓ ↗

Table 3.7: Average CLL over all target domain predicates.

from scratch, and SR2LR always has a significant advantage over MTAMAR. As expected, transfer to or from WebKB and the other two domains leads to less consistent gains and, in some cases, degradation. SR2LR is competitive also in terms of CLL, although in some cases, as discussed earlier, a model that gives significant advantages in AUC is at a disadvantage in CLL.

Focus on Specific Predicates: We have shown that, over all predicates in a domain, SR2LR can lead to significant gains in accuracy. Next, we study in greater detail the performance on the `WorkedFor` predicate in IMDB and `AdvisedBy` in UW-CSE, which, as argued earlier, require more data to be learned from scratch, and are best predicted by long-range clauses. The choice of `AdvisedBy` as the predicate to study in more detail is also motivated by the fact that it has been treated as the target predicate by several authors (e.g., Davis et al., 2007; Biba et al., 2008; Singla & Domingos, 2008). We picked `WorkedFor` because it corresponds to `AdvisedBy` in the IMDB domain.

We used the single-entity-centered instances from our experiments for the overall performance and introduced an additional baseline we call **SR-Only.**

SR-Only Baseline: Uses SR2LR to transfer only the short-range clauses, ignoring the long-range ones. This baseline is used to verify that transferring the long-range clauses is beneficial.

Statistically significant improvement (degradation) of SR2LR over SR-Only is indicated by a $\Uparrow$ ($\Downarrow$). As shown in Table 3.8, when transferring to IMDB from UW-CSE, SR2LR significantly outperforms all other methods. SR2LR also leads to significant gains in transfer from IMDB to UW-CSE, although in this case SR2LR is significantly better than SR-Only just on CLL, equaling its performance on AUC. Transferring from IMDB to UW-CSE is less beneficial than going in the opposite direction, from UW-CSE to IMDB, because several predicates in UW-CSE do not have analogs in IMDB while most of IMDB's predicates have a matching predicate in UW-CSE. As before, transfer from the more distantly related WebKB domain produces mixed results.

Source	MTAMAR	SR-only	Scratch	SR2LR
UW-CSE-manual	0.726	0.339	0.032	0.982 $\uparrow \Uparrow \nearrow$
UW-CSE-learned	0.024	0.215	0.032	0.239 $\uparrow \Uparrow \nearrow$
WebKB-learned	0.025	0.023	0.032	0.023 $\downarrow \swarrow$

Source	MTAMAR	SR-only	Scratch	SR2LR
IMDB-learned	0.010	0.030	0.008	0.030 $\uparrow \nearrow$
WebKB-learned	0.007	0.007	0.008	0.007 $\swarrow$

Table 3.8: AUC for `WorkedFor` in IMDB (top) and `AdvisedBy` in UW-CSE (bottom).

Source	MTAMAR	SR-only	Scratch	SR2LR
UW-CSE-manual	-0.084	-0.066	-6.488	-0.037 $\uparrow \Uparrow \nearrow$
UW-CSE-learned	-0.385	-0.695	-6.488	-0.727 $\downarrow \Downarrow \nearrow$
WebKB-learned	-0.728	-0.700	-6.488	-0.700 $\uparrow \nearrow$

Source	MTAMAR	SR-only	Scratch	SR2LR
IMDB-learned	-1.767	-0.295	-5.542	-0.280 $\uparrow \Uparrow \nearrow$
WebKB-learned	-0.757	-0.696	-5.542	-0.696 $\uparrow \nearrow$

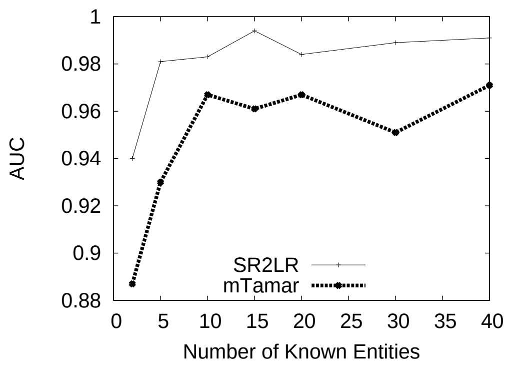
Table 3.9: CLL for `WorkedFor` in IMDB (top) and `AdvisedBy` in UW-CSE (bottom).

Increasing Numbers of Entities: In our final set of experiments, we compared the accuracy of SR2LR versus that of MTAMAR on `WorkedFor` and `AdvisedBy`, as information about more entities becomes available. To do this, we considered 5 distinct orderings of the constants of type person in each mega-example, and provided the first n to the systems, with n ranging from 2 to 40 in IMDB, where the smallest mega-example has 44 constants of type person and from 2 to 50 in UW-CSE, where the smallest mega-example has 56 such constants. Each point on the curves is the average over all training instances with that many known entities. The results in terms of AUC are shown in Figure 3.10. As can be seen, SR2LR maintains its effectiveness even as more data becomes available. Surprisingly, in UW-CSE MTAMAR's performance actually decreases as more entities become known. We conjecture that this is due to the fact that whereas SR2LR keeps all mappings that are supported by the data, MTAMAR picks the best mapping in terms of WPLL score for each source clause. As more entities become known, there are a larger number of possible relations among them. If the known entities are disconnected, however, MTAMAR does not observe many instances in which mappings of the long-range clauses are helpful and therefore rejects them in favor of mappings that produce short-range clauses (by mapping source predicates to the "empty" target predicate), for which there is growing support. SR2LR is not susceptible to this because it treats long-range and short-range clauses separately. This effect is not observed in the smaller IMDB domain where randomly chosen entities are much less likely to be disconnected.

This last set of experiments raises the interesting point of when, if at all, one should switch from SR2LR to MTAMAR, as the number of known entities grows. Our experiments provide indirect evidence that in some cases it might be better to use a simpler, less discriminating, measure to evaluate potential clause mappings.

3.3.2.1 Using Sources Learned with Original BUSL

Finally, we would like to compare the performance of SR2LR using source MLNs learned with the original BUSL to its performance using the sources from our main experiments. Tables 3.10 and 3.11 present a comparison between the performance of SR2LR from sources learned with the original BUSL algorithm to



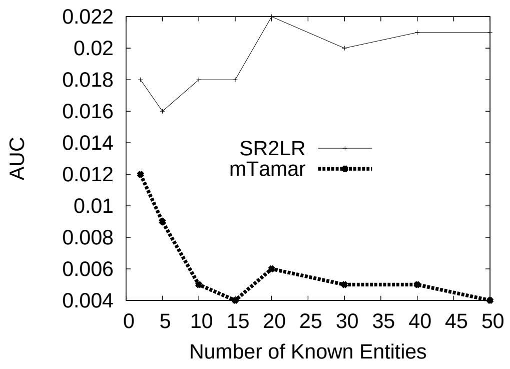


Figure 3.10: Accuracy on increasing amounts of data on `WorkedFor` (left) and `AdvisedBy` (right).

those learned with the slightly modified version of BUSL used in Section 3.3.2. Tables 3.12 and 3.13 show the performance on the `WorkedFor` and `AdvisedBy` predicates respectively. As can be seen, in some cases, the sources learned with the slightly modified BUSL, which perform worse than those of the original BUSL in the source domain, sometimes give better results when used for transfer.

Target	Source	SR2LR (modified BUSL)	SR2LR (original BUSL)
IMDB	UW-CSE	0.452	0.428
IMDB	WebKB	0.468	0.503
UW-CSE	IMDB	0.188	0.160
UW-CSE	WebKb	0.174	0.228
WebKb	IMDB	0.168	0.168
WebKb	UW-CSE	0.295	0.167

Table 3.10: Comparison in terms of AUC between the performance of SR2LR from sources learned with the modified versus original BUSL.

Target	Source	SR2LR (modified BUSL)	SR2LR (original BUSL)
IMDB	UW-CSE	-0.682	-0.816
IMDB	WebKB	-0.872	-0.609
UW-CSE	IMDB	-0.606	-0.839
UW-CSE	WebKb	-0.891	-0.618
WebKb	IMDB	-0.694	-0.693
WebKb	UW-CSE	-0.643	-1.687

Table 3.11: Comparison in terms of CLL between the performance of SR2LR from sources learned with the modified versus original BUSL.

Target	Source	SR2LR (modified BUSL)	SR2LR (original BUSL)
IMDB	UW-CSE	0.239	0.028
IMDB	WebKb	0.023	0.026
UW-CSE	IMDB	0.030	0.035
UW-CSE	WebKb	0.007	0.008

Table 3.12: Comparison in terms of AUC between the performance of SR2LR from sources learned with the modified versus original BUSL on the `AdvisedBy` predicate in UW-CSE and `WorkedFor` predicate in IMDB.

3.4 Summary

In this chapter, we presented two algorithms for transfer of MLN structure. The first one, RTAMAR, revises an MLN learned in a source domain and mapped to the predicates of the target domain in the case when a substantial amount of target-domain data is provided. The second algorithm, SR2LR, addresses the scenario when target-domain data is severely limited. Our experiments demonstrated that both of these algorithms lead to benefits in the accuracy and/or speed of learning.

Target	Source	SR2LR (modified BUSL)	SR2LR (original BUSL)
IMDB	UW-CSE	-0.727	-0.500
IMDB	WebKb	-0.700	-0.688
UW-CSE	IMDB	-0.280	-0.586
UW-CSE	WebKb	-0.696	-0.688

Table 3.13: Comparison in terms of CLL between the performance of SR2LR from sources learned with the modified versus original BUSL on the `AdvisedBy` predicate in UW-CSE and `WorkedFor` predicate in IMDB.

Chapter 4

MLN Structure Learning from Scratch

In Chapter 3 we presented algorithms for improving learning of an MLN via transfer of a model from a related source domain. In this chapter, we present a novel algorithm that aims at improving MLN structure learning from scratch by approaching the problem in a more bottom-up way. We call our algorithm BUSL for Bottom-Up Structure Learning (Mihalkova & Mooney, 2007).

4.1 BUSL Overview

As pointed out by Richardson and Domingos (2006), MLNs serve as templates for constructing Markov networks when different sets of constants are provided. Because the cliques of the ground Markov network are defined by the groundings of the same set of first-order clauses, the graph exhibits a high degree of redundancy where the same pattern is repeated several times, corresponding to each grounding of a particular clause.

Example 4.1.1. Considering Figure 2.4 (page 29) again, we observe that the pattern of nodes and edges appearing above the two `Credits` gliterals is repeated below them with different constants. In fact, this Markov network can be viewed as an instantiation of the template shown in Figure 4.1.

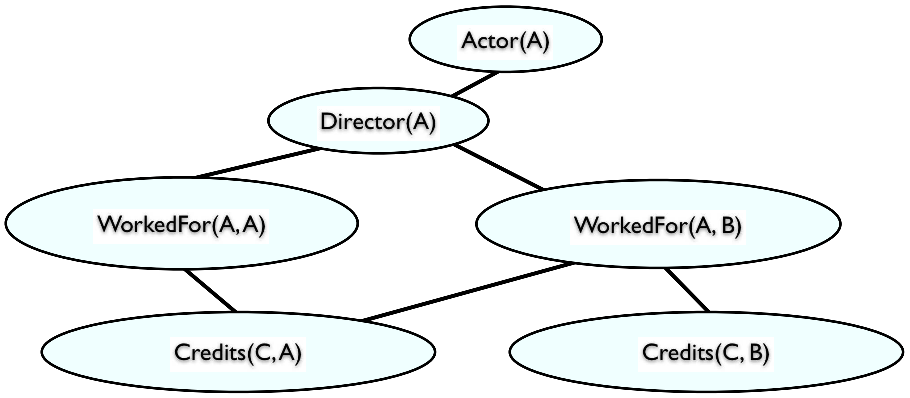


Figure 4.1: Example Markov Network Template

The basic idea behind BUSL is to learn MLN structure by first creating a Markov network template similar to the one shown in Figure 4.1 from the provided data. The nodes in this template are used as components from which clauses are constructed, and can contain one or more vliterals that are connected by a shared variable. We will call these nodes *TNodes* for template nodes. As in ordinary Markov networks, a TNode is independent of all other TNodes given its immediate neighbors. Recall from Section 2.3.1, that the Hammersley Clifford Theorem guarantees that we can specify any probability distribution compliant with the conditional independencies implied by a particular graph by using functions defined only over the cliques of the graph. In the case of MLNs where the functions are expressed as first-order logic rules, this implies that to learn the structure, the algorithm only needs to consider clause candidates that comply with the Markov network template. In other words, BUSL uses the Markov network template to restrict the search space for clauses only to those candidates whose literals correspond to TNodes that form a clique in the template.

Algorithm 4 Skeleton of BUSL

```
for each $P \in \mathcal{P}$ do
    Construct TNodes for predicate $P$ (Section 4.2.1)
    Connect the TNodes to form a Markov network template (Section 4.2.2)
    Create candidate clauses, using this template to constrain the search (Section 4.2.3)
end for
Remove duplicate candidates
Evaluate candidates using WPLL and add best ones to final MLN
```

The approach taken by BUSL follows the same philosophy as the graph-centric learners discussed in Section 2.3.2 where the algorithm first focuses on learning the conditional independencies among the variables before specifying the features that define the probability distribution. This is in stark contrast to KD, which takes a feature-centric approach and proceeds by directly learning the clauses of the MLN.

Algorithm 4 gives the complete skeleton of BUSL. Letting $\mathcal{P}$ be the set of all predicates in the domain, the algorithm considers each predicate $P \in \mathcal{P}$ in turn. A Markov network template is *automatically* constructed from the perspective of the current target predicate P. Template construction involves creating variablized TNodes, or components for clause construction, and determining the edges between them. Even though the template aids the search for clauses, it does not carry all the information about the MLN. Namely, it does not specify whether the vliterals participating in a clause are positive or negative, or precisely what clauses correspond to a given clique. For example, a three-node clique could correspond to one three-literal clause or to three two-literal clauses, etc. Information about the weights is also excluded. To search for actual clauses, we generate clause candidates by focusing on each maximal clique in turn and producing all possible clauses consistent with it. More specifically, these are all possible clauses of length 1 to $cliqueSize$ containing only members of the clique. We can then evaluate each candidate using the WPLL score (Kok & Domingos, 2005) (described on page 31). In the following section we give the details of each step.

4.2 BUSL Details

A Markov network template is created for each predicate in the domain in order to ensure that the relationships of all predicates are properly modeled. Below, we describe the process for the current target predicate P.

4.2.1 TNode Construction

TNodes contain conjunctions of one or more vliterals and serve as building blocks for creating clauses. Intuitively, TNodes are constructed by looking for groups of constant-sharing gliterals that are true in the data and variablizing them. Thus, TNodes could also be viewed as portions of clauses that have true groundings in the data. The process of TNode construction is inspired by relational pathfinding (Richards & Mooney, 1992), which we described in Section 2.2.3. The result of running TNode construction for P is the set of TNodes and a matrix M_P containing a column for each of the created TNodes and a row for each gliteral of P. Each entry $M_P[r][c]$ is a Boolean value that indicates whether the data contains a true grounding of the TNode corresponding to column c with at least one of the constants of the gliteral corresponding to row r. This matrix is used later to find the edges between the TNodes. Algorithm 5 describes how the set of TNodes and the matrix M_P are constructed. The algorithm uses the following definitions:

Definition 4.2.1. Two gliterals are *connected* if there exists a constant that is an argument of both of them. Similarly, two vliterals are connected if there exists a variable that is an argument of both of them.

Definition 4.2.2. A *chain* of literals of length l is a list of l literals such that for $1 < k \leq l$ the kth literal is connected to the $(k-1)$th via a previously unshared variable.

First, in line 1 the algorithm creates a *head* TNode that consists of a vliteral of P in which each argument is assigned a unique variable. This TNode is analogous to the head in a definite clause; however, note that our algorithm is not limited to constructing only definite clauses. Next, in lines 2 to 22 the algorithm considers each gliteral G_P of P in turn. This includes both the true and the false gliterals of P, where the true gliterals are those stated to hold in the data, while the rest are assumed to be false. A row of zeros is added to M_P for G_P, and the value corresponding to the head TNode is set to 1 if G_P is true and to 0 otherwise (lines 4-7). The algorithm then proceeds to consider the set $\mathcal{C}_{G_P}$ of all true gliterals in the data that are connected to G_P. For each $c \in \mathcal{C}_{G_P}$, it constructs each possible TNode based on c containing 1 to m vliterals. If a particular TNode was previously created, its value in the row corresponding to G_P is set to 1. Otherwise, a new column of zeros is added to M_P and the entry in the G_P row is set to 1 (lines 13-19). Thus,

Algorithm 5 Construct TNode Set

```
Input:  P: Predicate currently under consideration
    m: Maximum number of vliterals in a TNode
Output:  TNodeVector: Vector of constructed TNodes
    $M_P$: Matrix of Boolean values
Procedure:
 1: Make head TNode, headTN, and place it in position 0 of TNodeVector
 2: for each  (true or false) gliteral, $G_P$, of P do
 3:    Add a row of zeros to $M_P$
 4:    currRowIndex $=$ numRows($M_P$) $- 1$
 5:    if $G_P$ is true then
 6:       Set $M_P$[currRowIndex][0] $= 1$
 7:    end if
 8:    Let $\mathcal{C}_{G_P}$ be the set of true gliterals connected to $G_P$
 9:    for each c $\in \mathcal{C}_{G_P}$ do
10:       for each possible TNode of length 1 to m based on c do
11:          size $=$ current length
12:          newTNode $=$ CreateTNode(c, $G_P$, headTN, size) (Algorithm 6)
13:          position $=$ TNodeVector.find(newTNode)
14:          if position is not found then
15:             append newTNode to end of TNodeVector
16:             append a column of zeros to $M_P$
17:             position $=$ numColumns($M_P$) $- 1$
18:          end if
19:          Set $M_P$[currRowIndex][position] $= 1$
20:       end for
21:    end for
22: end for
```

each entry in M_P indicates whether the TNode corresponding to its column could be formed when considering the gliteral corresponding to its row.

Algorithm 6 shows the CreateTNode procedure. In line 1, the algorithm variablizes the current gliteral c connected to G_P by replacing the constants c shares with G_P with their corresponding variables from the head TNode. If the TNode size

Algorithm 6 CreateTNode

```
Input:  G_P: Current gliteral of P under consideration
        c: Gliteral connected to G_P on which this TNode is based
        headTN: Head TNode
        size: Number of vliterals in the TNode
Output: newTNode: The constructed TNode
Procedure:
 1: v = variablize c such that the constants shared with G_P are replaced with their
    corresponding variables from headTN and all others are replaced with unique
    variables
 2: Create newTNode containing v
 3: previousGliteral = c
 4: lastVliteralInChain = v
 5: while length(newTNode) < size do
 6:     c_1 = pick true gliteral connected to previousGliteral via a previously
        unshared constant
 7:     v_1 = variablize c_1 such that constants shared with G_P or previousGliteral
        are replaced with their corresponding variables from headTN or
        lastVliteralInChain and all others are replaced with unique variables
 8:     Add v_1 to newTNode
 9:     previousGliteral = c_1
10:     lastVliteralInChain = v_1
11: end while
```

is greater than 1, the algorithm enters the while loop in lines 5-11. In each iteration of this loop we extend the TNode with an additional vliteral that is constructed by variablizing a gliteral connected to the gliteral considered in the previous iteration so that any constants shared with the head TNode or with the previous gliteral are replaced with their corresponding variables.

Example 4.2.1. Suppose that for our example domain, we are given the database in Figure 4.2. Let $P = \texttt{Actor}$ and $m = 2$ (i.e. at most 2 vliterals per TNode). The head TNode is Actor(A). Figures 4.3 and 4.4 show the gliteral chains considered

Actor(brando) Director(coppola)
WorkedFor(brando, coppola)
Credits(godFather, coppola) Credits(godFather, brando)

Figure 4.2: Database used in Example 4.2.1. The listed gliterals are `true`; the rest are `false`.

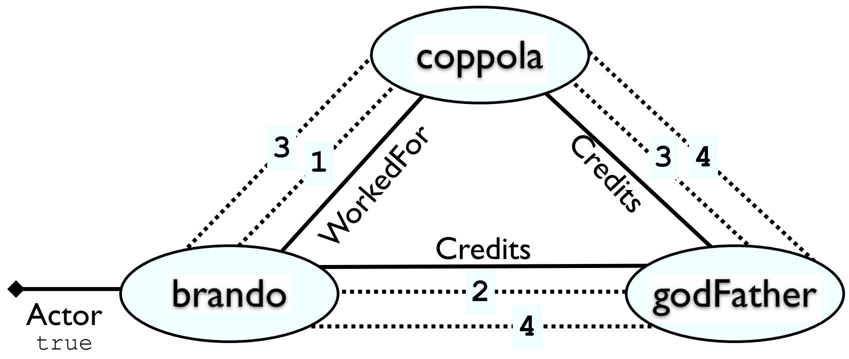


Figure 4.3: An illustration of the chains considered when constructing TNodes for `Actor(brando)`, which is a true gliteral. The solid edges show existing relations between the constants. The dashed edges indicate paths, where each path is numbered. Paths 1 and 2 have length one, and paths 3 and 4 have length two.

in the main loop (lines 2-22) of Algorithm 5 for each gliteral of P.

Let us first focus on the case when G_P is `Actor(brando)` shown in Figure 4.3. Connections 1 and 2 lead to the TNodes `WorkedFor(A, B)` and `Credits(C, A)` respectively. Connection 3, from `brando` to `coppola` via the `WorkedFor` edge and then to `godFather` via the `Credits` edge, gives rise to the 2-vliteral TNode `[WorkedFor(A, D), Credits(E, D)]`. Connection 4, which goes from `brando` to `coppola` via `godFather`, motivates the TNode `[Credits(F, A), Credits(F, G)]`. The following table lists the values in M_P at this point.

Actor(A)	WorkedFor(A, B)	Credits(C, A)	WorkedFor(A, D) Credits(E, D)	Credits(F, A) Credits(F, G)
1	1	1	1	1

Note that when constructing the TNodes, we replaced shared constants with the same variables, and constants shared with G_P with the corresponding variable from the head TNode.

We did not consider the chain `[Credits(godFather, brando)`, `WorkedFor(brando, coppola)]`. This chain is invalid because the shared constant, `brando`, has been shared previously (with the head TNode). We can use this example of an invalid chain to provide some intuition for the requirement that a chain can be extended only by sharing a previously unshared constant. Suppose that this restriction did not exist. Then we would form the TNode

```
[Credits(X1,A), WorkedFor(A, X2)]
```

However, we notice that the vliterals composing this new TNode are present, modulo variable renaming, in two separate TNodes found earlier (the second and third TNodes in the table above). Therefore, constructing this TNode has the effect of producing two-vliteral TNodes consisting of vliterals that already appear in single-vliteral TNodes.

Next, we consider Figure 4.4 that deals with the second iteration in which G_P is `Actor(coppola)`. Based on connection 5, we construct a new TNode `Director(A)` and from connection 6 the TNode `WorkedFor(H, A)`, which

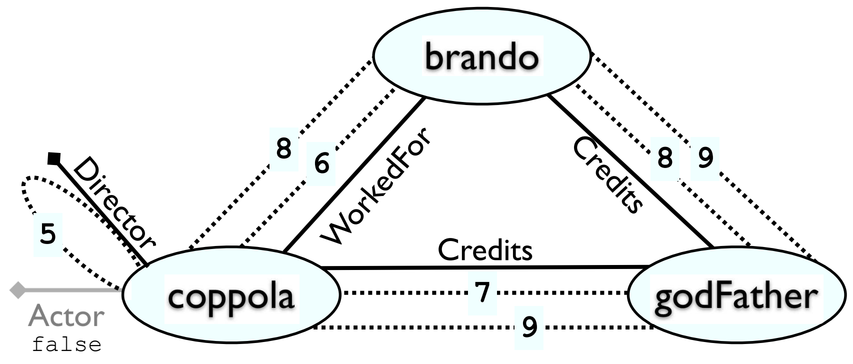


Figure 4.4: An illustration of the chains considered when constructing TNodes for `Actor(coppola)`, which is a false gliteral. The solid edges show existing relations between the constants. The dashed edges indicate paths, where each path is numbered. Paths 5, 6, and 7 have length one, and paths 8 and 9 have length two.

Actor(A)	WkdFor(A, B)	Credits(C, A)	Director(A)	WkdFor(D, A)	WkdFor(A, E), Credits(F, E)	Credits(G, A), Credits(G, H)	WkdFor(I, A), Credits(J, I)
1	1	1	0	0	1	1	0
0	0	1	1	1	0	1	1

Table 4.1: Final set of TNodes and their corresponding M_P matrix

differs from the `WorkedFor` TNode found earlier by the position of the variable A shared with the head TNode. An appropriate TNode for connection 7 (`Credits(C,A)`) already exists. Connection 8 gives rise to the two-vliteral TNode `[WorkedFor(I, A), Credits(J, I)]`. A TNode for connection 9, `[Credits(F, A), Credits(F, G)]` was constructed in the previous iteration. Table 4.1 lists the final set of TNodes.

If TNodes are restricted to consist of only a single vliteral, BUSL would construct only clauses whose literals all contain a shared variable (the one shared with the head TNode). Such clauses can be viewed as revolving around a single entity, represented by the shared variable. Because TNodes of length 2 introduce paths based on shared variables that do not appear in the head TNode, when such TNodes are allowed, BUSL can construct clauses that extend beyond the relations of a single entity. In general, larger values of m mean longer TNodes that could help build more informative clauses. However, a larger m also leads to the construction of more TNodes, thus increasing the search space for clauses. We used a conservative setting of $m = 2$. Note that this does not limit the final clause length to 2. To further reduce the search space, we require that TNodes with more than one vliteral contain at most one free variable (i.e. a variable that does not appear in more than one of the vliterals in the TNode or in the head TNode). We did not experiment with more liberal settings of these parameters but, as our experiments demonstrate, these values worked well in our domains.

TNode construction is very much in the spirit of bottom-up learning. Rather than producing all possible vliterals that share variables with one another in all possible ways, the algorithm focuses only on vliterals for which there is a true gliteral in the data. Thus, the data already guides and constrains the algorithm. This is related to bottom-up ILP techniques such as least-general generalizations (LGG) and inverse resolution (Lavrač & Džeroski, 1994). However, as opposed to LGG, our TNode construction algorithm always uses the generalization that leads to completely variablized TNodes and unlike inverse resolution, the process does not lead to the creation of complete clauses and does not use any logical inference algorithms like resolution.

The procedure for constructing TNodes is also very similar to relational pathfinding (RPF) (Richards & Mooney, 1992), described on page 17. Like RPF, it is based on searching for paths in the relational graph of the data. However, unlike RPF, these paths do not attempt to connect the constants of a gliteral of the target predicate. Whereas in RPF the goal is to discover ways of proving true instances of the target predicate, the goal of TNode construction is to discover features that can be effective clause building blocks.

4.2.2 Adding the Edges

Once TNodes are constructed, we can search through the space of all possible clauses composed from them. This search space is already smaller than the one considered by KD because the algorithm uses only combinations of vliterals that contain at least one true grounding in the data. Nevertheless, the number of possible clauses may still be prohibitively large. Moreover, as discussed in Section 4.1, an exhaustive search is not necessary. Thus we proceed to complete the template construction, by finding which TNodes are connected by edges. For this purpose, it is useful to recall that the templates represent variablized analogs of Markov networks. Finding the edges can therefore be cast as a Markov network structure learning problem where the TNodes are the nodes in the Markov network and the matrix M_P provides training data. At this point, any Markov network learning algorithm can be employed. We chose the Grow-Shrink Markov Network (GSMN) algorithm by Bromberg et al. (2006), which we described in Section 2.3.2, because it is simple but effective. Our choice was also motivated by the fact that GSMN takes a graph-centric approach to the problem, which means that it learns just the structure of the Markov network, without any weights, which are unnecessary in our case.

4.2.3 Search for Clauses

Because the clauses in an MLN define functions over the cliques in the ground MLN, we should only construct clauses from TNodes that form cliques in the Markov network template. In other words, any two TNodes participating together in a clause must be connected by an edge in the template. The head TNode is required to participate in every candidate. Each clause can contain at most one multiple-literal TNode and at most one TNode that contains a single non-unary literal. These further restrictions on the clause candidates are designed to decrease the number of free variables in a clause, thus decreasing the size of the ground MLN during inference, and further reducing the search space. Complying with the above restrictions, we consider each clique in which the head TNode participates and construct all possible clauses whose length is less than the size of the clique by forming disjunctions from the literals of the participating TNodes with all possible negation/non-negation combinations.

After template creation and clause candidate generation are carried out for each predicate in the domain, duplicates are removed and the candidates are evaluated using the WPLL score (Kok & Domingos, 2005), described on page 31. Recall that in order to compute this score, one needs to assign a weight to each clause. Weight learning is performed using L-BFGS, also used by Richardson and Domingos (2006) and also used in KD. After all candidates are scored, they are considered for addition to the MLN in order of decreasing score. To reduce overfitting and speed up inference, only candidates with weight greater than *minWeight* are considered. Candidates that do not increase the overall WPLL of the currently learned MLN are discarded.

4.3 Experimental Setup

We compared the performance of BUSL to that of KD in the same three relational domains—IMDB, UW-CSE, and WebKB—that we described in Section 3.2.2. It is important to note that our results on the UW-CSE dataset are not comparable to those presented by Kok and Domingos (2005) because due to privacy issues we only had access to the published version of this data, which differs from the original (Personal communication by Stanley Kok).

As in Chapter 3, we measured the performance in terms of the AUC and CLL and generated learning curves using a leave-1-mega-example-out approach. The parameter settings for running KD from scratch were identical. As before, all timing runs within the same experiment were carried out on the same dedicated machine. We implemented BUSL as part of the Alchemy package (Kok et al., 2005). We set BUSL's $minWeight = 0.5$ for all experiments and observed that the operation of the algorithm is not very sensitive to other settings of this parameter. Even though both BUSL and KD have a parameter called $minWeight$, they use it in different ways and the same value is therefore not necessarily optimal for both systems. The p-value for the χ^2 test used by GSMN was set to 0.85 and was not further optimized.

4.4 Experimental Results

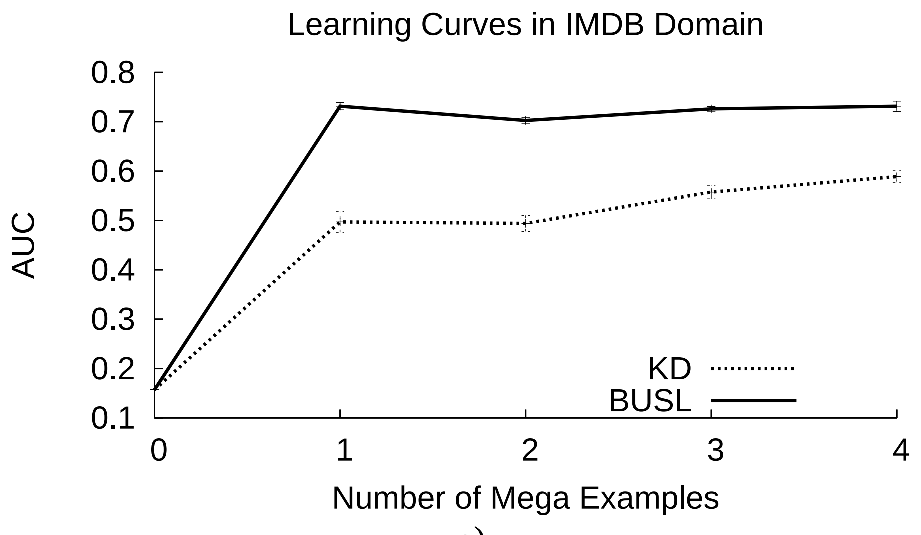
Figures 4.5-4.7 show learning curves in the three domains. BUSL improves over the performance of KD in all cases except for one point in terms of AUC.

Figure 4.6 additionally plots the AUC and CLL for a system that performs weight learning over the knowledge base provided as part of the UW-CSE dataset (Hand-KB). Hand-KB was generated by asking volunteers to express in first-order logic general knowledge about academia (Richardson, 2004). In terms of AUC, this system's performance is significantly worse than that of BUSL, and in terms of CLL, it performs as well as BUSL.

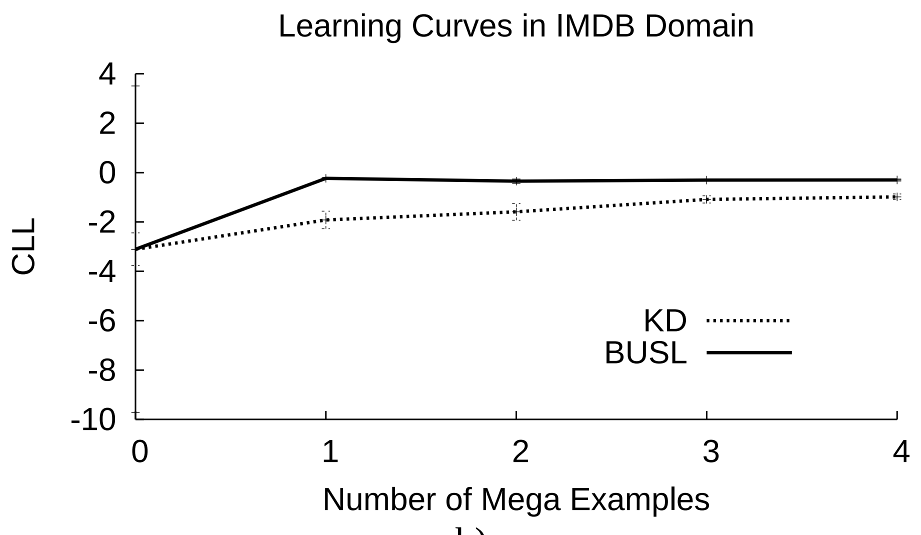
In Figure 4.7, we observe that even though KD is improving its performance in terms of AUC, its CLL score decreases. This is most probably due to the extremely small relative number of true gliterals in the domain in which the CLL can be increased by simply predicting `false` for each query.

Another observation that requires explanation is that the learners improve by only tiny amounts, if at all, after the first point on the learning curve. This occurs because in our experience, for both learners, additional data improves only the WPLL estimate (and thus the evaluation of new clause candidates) but does not have a great effect on the clauses that are proposed. In particular, in BUSL candidates are based on the dependencies among the TNodes, and new data introduces few new such dependencies. This, however, may not be the case in other domains.

Figures 4.5-4.7 give an idea of how the learners perform over all the predicates of the domain. It is also interesting, however, to see the performance of the

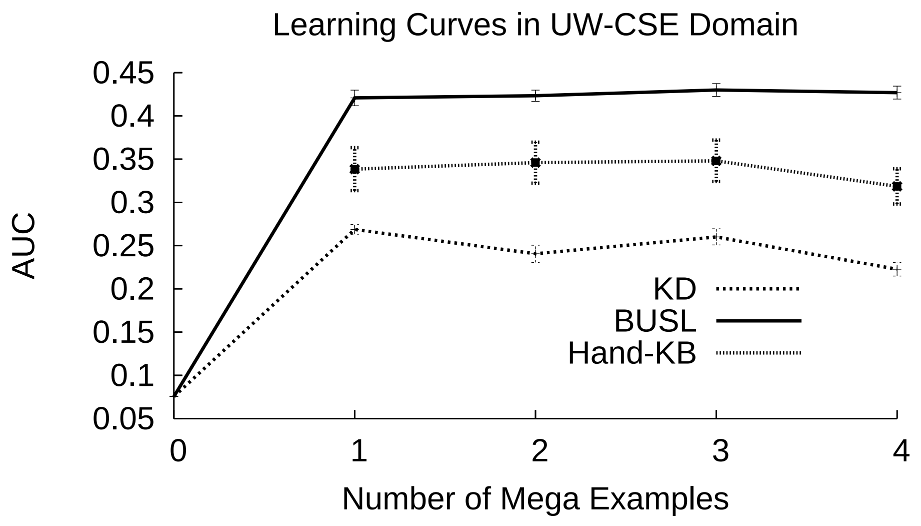


a)

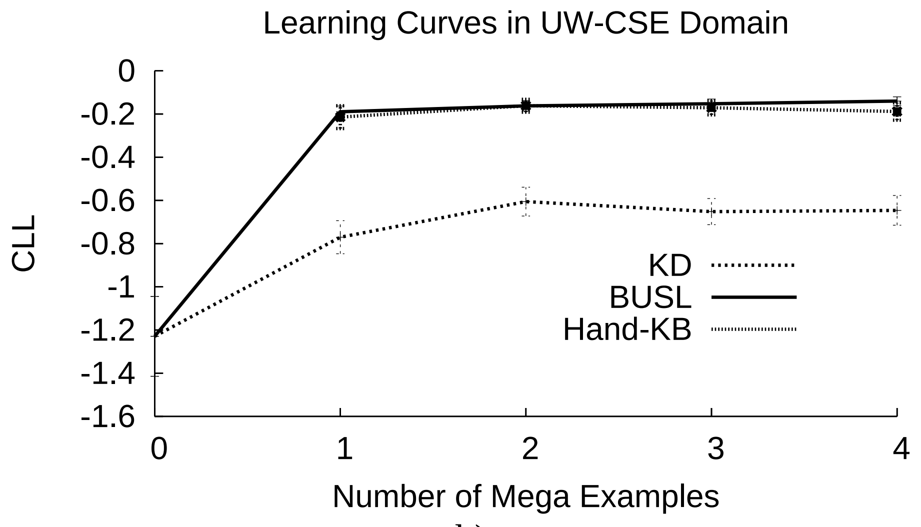


b)

Figure 4.5: Accuracy in IMDB domain. a) AUC b) CLL

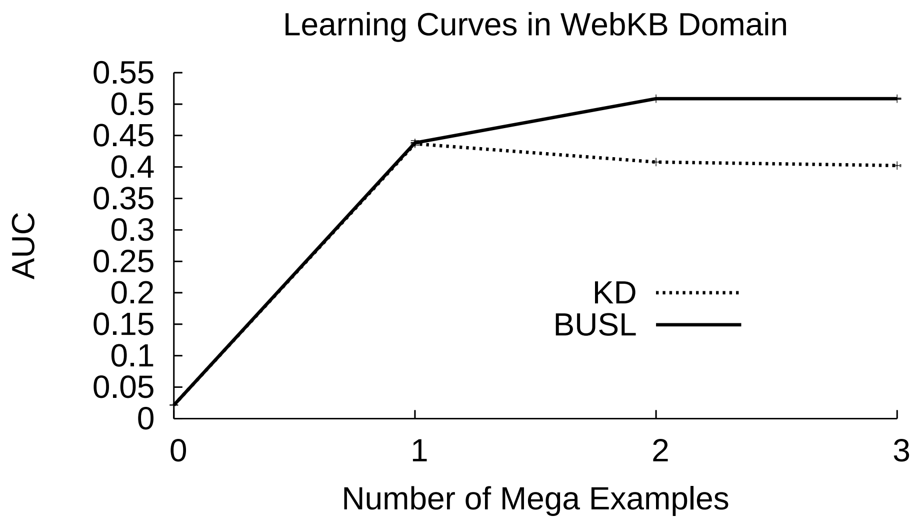


a)

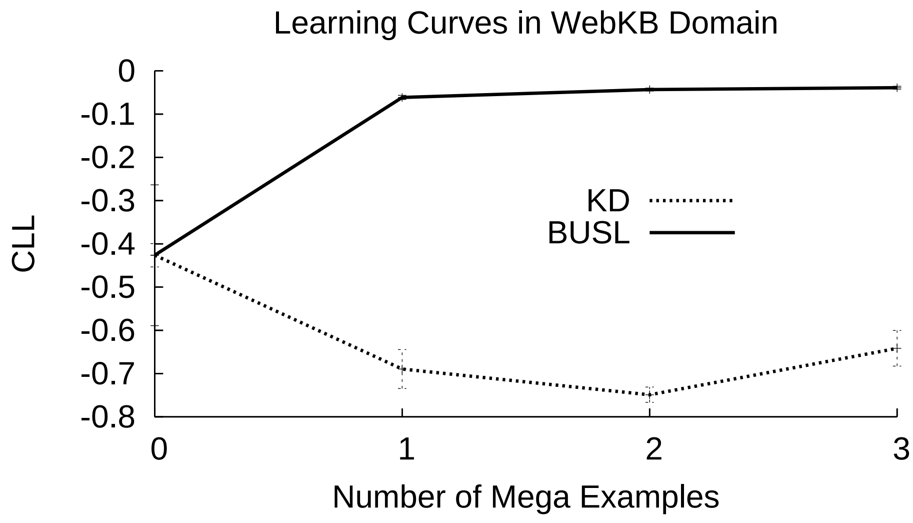


b)

Figure 4.6: Accuracy in UW-CSE domain. a) AUC b) CLL



a)



b)

Figure 4.7: Accuracy in WebKB domain. a) AUC b) CLL

Predicates	CLL BUSL	CLL KD	AUC BUSL	AUC KD
director	-0.24±0.12	-1.44±0.12	0.91±0.03	0.51±0.01
actor	-0.01±0.00	-0.59±0.08	1.00±0.00	0.88±0.01
movie	-1.66±0.17	-2.42±0.25	0.27±0.00	0.19±0.00
gender	-0.69±0.05	-3.33±0.33	0.48±0.01	0.36±0.00
workedUnder	-0.07±0.00	-0.24±0.02	0.26±0.00	0.10±0.00
genre	-0.18±0.05	-1.10±0.04	0.60±0.05	0.34±0.02
samePerson	-0.03±0.00	-0.03±0.01	1.00±0.00	0.89±0.01
sameMovie	-0.04±0.00	-0.11±0.03	1.00±0.00	0.99±0.00
sameGenre	-0.05±0.00	-0.44±0.23	0.80±0.00	0.63±0.04
sameGender	-0.04±0.00	-0.14±0.07	1.00±0.00	0.99±0.01

Table 4.2: Per-predicate results from last point on learning curve in IMDB

systems for each predicate in the domains individually. Tables 4.2-4.4 show these results for AUC and CLL for the last point on the learning curves. Note that the performance for some of the predicates, such as `TaughtBy` in UW-CSE is extremely low. This is due to the fact that, given the information provided during testing, it is impossible to reliably predict the value of these predicates.

Table 4.5 shows the average training time over all learning runs for each system, and the average number of candidate clauses each learner constructed and evaluated over all runs. As can be seen, BUSL constructed fewer candidates and trained much faster than KD. BUSL spends the main portion of its training time on computing the WPLL score of the generated candidates. This process takes longer in domains like WebKB that contain a great number of constants. On the other hand, we expect BUSL's savings in terms of number of generated candidates to be greater in domains, such as UW-CSE, that contain many predicates because the large number of predicates increases the number of candidate clauses gener-

Predicates	CLL BUSL	CLL KD	AUC BUSL	AUC KD
taughtBy	-0.02±0.00	-0.03±0.00	0.01±0.00	0.00±0.00
courseLevel	-0.82±0.08	-2.95±0.37	0.48±0.03	0.28±0.01
position	-0.16±0.03	-1.33±0.08	0.33±0.03	0.09±0.02
advisedBy	-0.04±0.01	-0.12±0.01	0.02±0.00	0.00±0.00
projectMember	-0.02±0.00	-0.01±0.01	0.00±0.00	0.00±0.00
phase	-0.35±0.03	-0.75±0.13	0.32±0.01	0.26±0.01
tempAdvisedBy	-0.02±0.00	-0.09±0.01	0.01±0.00	0.00±0.00
yearsInProgram	-0.22±0.04	-0.37±0.04	0.16±0.02	0.10±0.01
tA	-0.03±0.00	-0.02±0.00	0.00±0.00	0.00±0.00
student	-0.06±0.02	-1.58±0.10	1.00±0.00	0.59±0.03
professor	-0.07±0.05	-1.51±0.08	0.98±0.01	0.16±0.03
samePerson	-0.03±0.00	-0.06±0.01	1.00±0.00	0.79±0.00
sameCourse	-0.04±0.00	-0.29±0.06	1.00±0.00	0.41±0.00
sameProject	-0.04±0.00	-0.38±0.11	1.00±0.00	0.60±0.00
publication	-0.18±0.02	-0.20±0.02	0.10±0.01	0.05±0.00

Table 4.3: Per-predicate results from last point on learning curve in UW-CSE

ated by KD. These considerations explain why the smallest improvement in speed is achieved in WebKB that contains the least number of predicates and the greatest number of constants. The greatest speed-up is in IMDB where BUSL created the smallest number of candidates, and each candidate could be evaluated quickly because of the small number of constants in this domain.

Based on the much smaller number of candidate clauses considered by BUSL, one might expect a larger speed-up. Such a speed-up is not observed because of optimizations within Alchemy that allow fast scoring of clauses for a fixed structure of the MLN. Because KD evaluates a large number of candidates with a fixed structure, it can take advantage of these optimizations. On the other hand, after initially scoring all candidates, BUSL attempts to add them in decreasing or-

Predicate	CLL BUSL	CLL KD	AUC BUSL	AUC KD
student	-0.01±0.00	-0.81±0.09	1.00±0.00	0.93±0.00
samePerson	-0.02±0.00	-0.01±0.00	0.99±0.00	0.88±0.01
faculty	-0.02±0.00	-2.78±0.13	1.00±0.00	0.56±0.00
project	-0.13±0.01	-0.17±0.02	0.03±0.00	0.02±0.00
courseTA	-0.03±0.00	-0.03±0.00	0.01±0.00	0.01±0.00
courseProf	-0.03±0.00	-0.04±0.01	0.02±0.00	0.01±0.00

Table 4.4: Per-predicate results from last point on learning curve in WebKB

	Training time			# candidates	
Dataset	BUSL	KD	Speed-up	BUSL	KD
IMDB	4.59	62.23	13.56	162	7558
UW-CSE	280.31	1127.48	4.02	340	32096
WebKB	272.16	772.09	2.84	341	4643

Table 4.5: Average training time in minutes, average speed-up factor, and average number of candidates considered by each learner.

der of score to the MLN, thus changing the MLN at almost each step, which slows down the scoring of the structure.

Finally, we checked the importance of adding the edges in Section 4.2.2. This step can, in principle, be avoided by simply producing a fully connected Markov network template. Recall that the goal of this step is to decrease the number of vliterals that could participate together in a clause. In Table 4.6 we show statistics on the number of TNodes constructed by the algorithm in each of the domains, as well as the proportion of TNodes that end up in the Markov blanket of the head TNode. As can be seen, the number of neighbors of the head TNode in the Markov network template is dramatically smaller than the total number of TNodes

Data set	IMDB	UW-CSE	WebKB
Average number of TNodes constructed	31.44	70.70	18.83
Average proportion of TNodes in MB of head TNode	0.12	0.14	0.22
Maximum number of TNodes constructed	56	144	28
Maximum size of MB of head TNode	17	41	15

Table 4.6: Statistics on the average number of TNodes constructed, the average proportion of TNodes that appear in the Markov blanket of the head TNode, the maximum number of TNodes constructed, and the maximum Markov blanket size, over the predicates in all learning runs in each domain.

discovered. This naturally leads to a smaller number of candidate clauses that need to be considered.

As mentioned in Section 2.4.2 (page 31), at the time of writing of this manuscript, Kok and Domingos (2009) have just introduced LHL, a new algorithm for MLN structure learning, which, like BUSL, embraces a bottom-up perspective. Because LHL performs relational pathfinding (Richards & Mooney, 1992) on a lifted hypergraph, it is able to search for longer paths than BUSL in a reasonable amount of time, which enables LHL to achieve excellent performance on large datasets, such as Cora (Bilenko & Mooney, 2003). As reported by Kok and Domingos (2009), using a slightly different experimental set-up from ours, LHL has accuracy comparable to that of BUSL on the UW-CSE dataset, but shorter training time; on IMDB, it outperforms BUSL in terms of accuracy, but takes longer to train. Results are not reported for the WebKB dataset.

Chapter 5

Using MLNs to Resolve Ambiguous Web Queries

In Chapter 3, we considered approaches for overcoming one way in which training data may be limited, when information only about a small group of entities is available. In this chapter, we demonstrate how through the use of relational information we can overcome a limitation on the amount of entity-specific data that is provided. In other words, here we assume that very little is known about each entity and we develop an approach that bases its predictions on relations among the entities. We focus on a particular application, Web query disambiguation, in which the task is to determine the intent of a search-engine user when she enters a potentially ambiguous query. We consider a more privacy-aware setting in which the only information available about any particular user is that captured in a short search session of 4–6 previous searches on average.

5.1 Motivating Web Query Disambiguation from Short Sessions

Personalizing a user's Web search experience has become a vibrant area of research in recent years. One of the most actively researched topics in this area is Web query disambiguation, or automatically determining the intentions and goals of a user who enters an ambiguous query. This is not surprising, given the frequency of ambiguous searches and the unwillingness of users to enter long and descriptive queries. For example, Jansen and Spink (2006) found that about 30% of search queries, submitted to several engines, consisted of a single word. Furthermore, Sanderson (2008) reports that anywhere between roughly 7% and 23% of the queries frequently occurring in the logs of two search engines are ambiguous, with the average length of ambiguous queries being close to one.

Ambiguity exists not only in cases such as the all-too-familiar "jaguar" example (which can be a cat, car, or operating system), but also in searches that do not appear ambiguous on the surface. Queries that are commonly considered unambiguous often become ambiguous as a result of the wealth of Web sources, which examine different aspects of a given topic. For example, as we observed in our data, a search for "texas"[1] may be prompted by at least two different kinds of intentions. In one session, a user who had first searched for "george w. bush" proceeded to search for "texas" and selected `www.tea.state.tx.us`, thus indicating an interest in Texas government agencies. In another session, the user intended to learn about travel to Texas because repeated searches for "georgia travel" were followed by a search for "texas" and a click to `www.tourtexas.com`. This indicates that even a query, such as "texas" that normally refers to a single entity, may become ambiguous.

Most approaches to Web query disambiguation leverage a user's previous interactions with the search engine to predict her intentions when entering an am-

[1]We write these queries in lower-case because this is how they were typed by the searchers in our data set.

biguous query. Typically, the actions of each user are logged over long periods of time (e.g., Sugiyama, Hatano, & Yoshikawa, 2004; Sun, Zeng, Liu, Lu, & Chen, 2005; Dou, Song, & Wen, 2007). While techniques that assume the availability of long search histories *for each user* are applicable in some situations, in many cases such approaches may raise privacy concerns and may be difficult to implement for pragmatic reasons. After the release of AOL query log data allowed journalists to identify one user based on her searches (Barbaro & Zeller, 2006), many people have become especially wary of having their entire search histories recorded by search engines. This has led to increased interest in the ethical issues surrounding user data collection (e.g., Conti, 2006), and the appearance of search engines that expressly do not store any user activity information, such as Cuil.[2]

However, in order to determine user intent when typing an ambiguous query, at least some information must be available about the user. We present an approach that bases its predictions only on short glimpses of user search activity, captured in a brief search session (Mihalkova & Mooney, 2009a). Our approach relates the current search session to previous *short* sessions of *other* users based on the search activity in these sessions. Crucially, our approach does *not* assume the availability of user identifiers of any sort (i.e. IP addresses, login names, etc.) and thus such information, which could allow user searches to be tracked over long periods of time, does not need to be recorded when our approach is used.

As an example, consider the query "scrubs," which could refer either to the

[2] `http://www.cuil.com/`

—Search Session 1—		
98.7 fm	→	`www.star987.com/main.html`
kroq	→	`www.kroq.com/`
scrubs	→	`scrubs-tv.com`
—Search Session 2—		
huntsville hospital	→	`www.huntsvillehospital.org`
ebay.com	→	`ebay.com`
scrubs	→	`www.scrubs.com`

Table 5.1: Two sessions in which the users searched for the query “scrubs.”

popular television show or to a type of medical uniform. Table 5.1 juxtaposes the users’ actions in two sessions. The sessions are short, with each containing only two searches preceding the ambiguous query; nevertheless, this short glimpse of the users’ actions is sufficient to provide an accurate idea of the users’ intentions because by examining historical data, one may discover that people who search for radio stations are probably “ordinary” users and would therefore be interested in the television show. On the other hand, by relating Session 2 to sessions of other users who searched for medical-related items, we may be able to predict that the second user has more specialized interests.

Our proposed setting is appealing also from a pragmatic standpoint because it does not require search engines to store, manage, and protect long user-specific histories. Identifying users across search sessions is another difficulty arising from methods based on long user-specific search histories. One possibility, to require users to log in before providing personalized search, may be cumbersome. The alternative of using as an identifier the IP address of the computer from which the search was initiated is also unsatisfactory, especially in cases when entire organizations share the same IP address or when all members of a household search from the same computer. Disambiguation techniques that explicitly do not use such identifiers and instead rely only on information from brief sessions avoid such difficulties.

When so little is known about a searcher, the problem of query disambiguation becomes very challenging. In fact, it has previously been argued that "it is difficult to build an appropriate user profile even when the user history is rich" (Dou et al., 2007). We develop an approach that successfully leverages the small amount of information about a user captured in a short search session to improve the ranking of the returned search results. Our approach uses MLNs to exploit the relations between the session in which the ambiguous query is issued and previous sessions.

SRL techniques are appealing for the problem of Web query disambiguation for two main reasons. First, the data is inherently relational—there are several types of entities: queries, clicked URLs, and sessions, which relate to each other in a variety of ways, e.g., two sessions may be related by virtue of containing clicks to the same URLs or searches for similar queries; queries may be related by sharing words or by being followed by clicks to the same URLs, and so on. SRL techniques allow us to learn *general* models of the ways in which the various types of entities interact, thus overcoming the problem that not much may be known about any particular entity, i.e. a particular URL. Second, data recording human interactions with a search engine is likely to be noisy. SRL models allow for probabilistic inference, helpful when reasoning from noisy data.

Before we describe the details of our approach, we discuss some related work.

5.2 Related Work

Web query disambiguation and personalized search are important problems, and have been studied under a variety of settings and assumptions. We review some of this work and draw distinctions between existing research and the work presented in this chapter.

5.2.1 Web Search Personalization

An early personalization techniques was developed by Fitzpatrick and Dent (1997). To disambiguate a query, their approach uses records of similar past queries over all users in order to include additional search terms in the original query, thus narrowing down the search. Unlike these authors, we are interested in re-ordering the results returned by the search engine rather than modifying that set by providing additional search terms.

Several authors have proposed techniques addressing the case where, for each particular user, a relatively long history of that user's interactions with the search engine is available. Sugiyama et al. (2004) present a personalization method that builds a user preference model by modeling separately the long-term and "today's" user interests. The user profile is viewed as a weighted average of these two components. In addition to relying on long-term records of user activity, their approach also uses the content of browsed web pages when constructing user profiles. In contrast, we are interested in a more light-weight approach that does not necessarily use page content. Sun et al. (2005) use spectral methods to perform personalization by organizing the data into a three-dimensional tensor comprised of users, queries, and clicked pages. In a related vein, Sun, Wang, Shen, Zeng, and Chen (2006) extended co-clustering (Dhillon, Mallela, & Modha, 2003) to work with three-dimensional tensors and simultaneously clustered users, queries, and pages. Because of the sparsity of the data, these tensor-based methods are unlikely to be effective in the case we study, where each user clicks on only a few pages and enters only a handful of queries.

A comprehensive empirical study of several Web search personalization techniques is presented by Dou et al. (2007). These techniques also use longer-term histories (up to 12 days) of the same user. The authors find that the best-performing methods are based on the intuition that the Web pages most relevant to a user are those clicked frequently in the past by that user or by related users, where user similarity is measured by estimating user membership in a pre-defined set of categories. Such a strategy is unlikely to work in our setting because the sessions in our data represent one-time interactions that usually do not contain repeated clicks to the same URL. Joachims (2002) and Radlinski and Joachims (2005) use a clever method for deriving constraints about user preferences by observing whether or not the user clicked on or skipped over particular search results. These preferences are then used to train a system for ranking search results according to user's preferences. Another related project (Teevan, Dumais, & Horvitz, 2005) relies on sensitive user information to personalize Web search by constructing a user profile from long-term observations on the user's activities, ranging from browsing history to e-mail. In general, all previous work discussed in the last two paragraphs makes the assumption that long-term information about *each user* is available. In contrast, we study the setting where personalization is performed based on records of very short interactions with the search engine.

To the best of our knowledge, the only previous work that targets query disambiguation from short sessions is that of Almeida and Almeida (2004) in which users are identified as belonging to a set of communities in order to determine their interests. The authors experimented with data from online bookstore search sites for computer science literature, and their approach is tailored for situations when user interests fall into a small set of categories, organizing users into 10 communities. While in a more restricted application of search, such as specialized book search, this small number of communities may be sufficient to model different aspects of user interests, when, as in our case, the goal is to disambiguate queries in a general-purpose search engine, a small number of communities is likely to be insufficient to effectively model the variety of user interests, and allowing for more communities may be prohibitively costly.

Privacy-aware Web personalization has been addressed by Krause and Horvitz (2008), whose method considers the privacy cost of a particular piece of user information and explicitly models the improvement in personalization versus the cost of the information that was used. While the ability to trade off performance with cost is highly desirable, their method relies on more information about the user than is available in our setting.

5.2.2 Learning to Rank

Learning for Web query disambiguation is also related to work on learning to rank (e.g., Burges et al., 2005). The latter task is to induce a model that produces good rankings of all the results relevant to a query, without targeting the specific interests of the current user. Query disambiguation can be viewed as auxiliary to this process, where we take the most relevant results, as determined by the general ranker, and re-order them for each user to better target the interests of that user. As in Web query disambiguation, models that incorporate implicit user feedback can lead to better results (e.g., Agichtein, Brill, & Dumais, 2006).

5.2.3 Determining User Intentions

Query disambiguation is also related to determining user goals and intentions. One of the earliest systems is Letizia (Lieberman, 1995), which operates on a client machine and observes the browsing behavior of a user. Upon request, Letizia can provide a ranking of the hyperlinks in a page based on its predictions of the user's interest. Another early system (Lesh & Etzioni, 1995) determines the goal of a user from an observed sequence of actions. These early approaches, however, do not incorporate a learning component.

The TaskPredictor (J. Shen, Li, Dietterich, & Herlocker, 2006) learns to predict the current task of a user based on the properties of the currently open window, or of an arriving e-mail message. Because training this system requires potentially sensitive information, such as e-mail and active documents, it is intended to be run on the user's local machine. In Web personalization, it is frequently necessary to determine whether a user issues a query to a general search engine with a particular goal in mind, such as job search, product search, or restaurant search. In this way, the search engine can deploy a service that was especially developed for that task. Query intent is resolved by classifying each query according to whether it indicates general or special interest (e.g., D. Shen, Sun, Yang, & Chen, 2006; Li, Wang, & Acero, 2008).

5.2.4 Producing Diverse Result Sets

Orthogonal to disambiguation is the issue of producing a diverse set of documents for a given query. Recent work in this area includes that of Chen and Karger (2006), whose technique ranks results so as to cover as many different aspects of interest as possible, and that of Yue and Joachims (2008) who propose a technique based on the structural SVM framework. A related area is that of clustering search results in groups of common topic. For example, Wang and Zhai (2007) use search log data to learn useful aspects of queries in order to cluster them. The ability to disambiguate user intent complements these contributions because it would allow the most relevant cluster, or the most relevant results from a diverse set, to be placed ahead of all others on the search page.

5.2.5 Collaborative Filtering

Our proposed approach is also related to work in collaborative filtering where the goal is to suggest items that would be of interest to a user, based on that and other users' previous preferences. Early comparative studies of collaborative filtering algorithms include (Breese et al., 1998; Herlocker et al., 1999). More recently, Popescul et al. (2001) and Melville et al. (2002) proposed approaches that combine collaborative and content-based information in forming recommendations. However, these approaches have not been applied to personalizing Web search.

5.3 Proposed Approach

Our general approach follows that of previous applications of MLNs to specific problems, (e.g., Poon & Domingos, 2007): we hand-coded the structure of the model as a set of first-order formulae and learned weights for these formulae from the data. This approach is also analogous to that commonly pursued in the probabilistic graphical model literature, where the dependencies among the variables of a graphical model are manually specified and then parameters that pin down the exact probability distribution are learned from the data. The advantage of using MLNs, however, is that they come with effective general-purpose learning and inference algorithms; thus one does not need to re-derive specialized inference techniques for every new model.

The key idea behind our approach is to relate the current, *active*, session A in which an ambiguous query Q is issued to previous, *background*, sessions from historical data, where it is assumed that both the active session and the background sessions are short. Sessions are related by sharing various types of information. We define the following predicates to capture these relationships. Since every training/testing example refers to a single (Q,A) pair, A and Q are implicit in the example and do not need to appear as arguments of the predicates.

- `Result`$(\texttt{R})$: `R` is a search result for Q.
- `ChoseResult`$(\texttt{S}, \texttt{R})$: Background session `S` clicked on `R` after searching for Q.
- `ClickOn`$(\texttt{R})$: User in session A clicks on result `R` in response to the search for Q.
- `SharesClick`$(\texttt{S}, \texttt{D})$: Background session `S` and A share a click to URL with hostname `D`.
- `SharesKeywordBtwnClicks`$(\texttt{S}, \texttt{K})$: Background session `S` and A share a keyword `K`, found in the hostnames of clicked URLs in each of the sessions.
- `SharesKeywordBtwnClickAndSearch`$(\texttt{S}, \texttt{K})$: Background session `S` and A share a keyword `K`, found in the hostname of a clicked URL in A and a search in `S`.
- `SharesKeywordBtwnSearchAndClick`$(\texttt{S}, \texttt{K})$: Background session `S` and A share a keyword `K`, found in a search in A and the hostname of a clicked URL in `S`.
- `SharesKeywordBtwnSearches`$(\texttt{S}, \texttt{K})$: Background session `S` and A share a keyword `K` that appeared in searches in both sessions.
- `ClicksShareKeyword`$(\texttt{R}, \texttt{D}, \texttt{K})$: Keyword K appears in the hostname of both result R and previous click D from session A.

- $\texttt{ClickAndSearchShareKeyword}(\texttt{R}, \texttt{S}, \texttt{K})$: Keyword K appears in the hostname of result R and in previous search query S from session A.

Figure 5.1 illustrates the predicates from the above set that are used to relate two sessions. The last two predicates capture information local to the active session. In the active session A, only the clicks and searches temporally *preceding* Q are used. For the predicates in which a keyword relates two sessions, we used only those keywords that appeared at least 100 times (corresponding to removing keywords that appeared less than 0.00083% of the time) and at most $10,000$ times (corresponding to removing the top 61 most popular keywords) over the training portion of our data set. This was done in order to avoid rare or misspelled keywords and to make the size of the data more manageable by excluding uninformative overly-common ones. We did not experiment with other cut-off values. We describe how the set of keywords is formed and how keywords are extracted from URLs in Section 5.4.

The goal is to predict the $\texttt{ClickOn}(\texttt{R})$ predicate, given as evidence the values of the remaining ones. The search results available for a given query are then ranked by the predicted probability that the user will choose to click on each of them.

5.3.1 Model Structure

This section describes the formulae used in our MLN models.

Collaborative Formulae: The collaborative formulae, shown in lines 1-5 of Table 5.2, draw inferences about the interests of the active user based on the

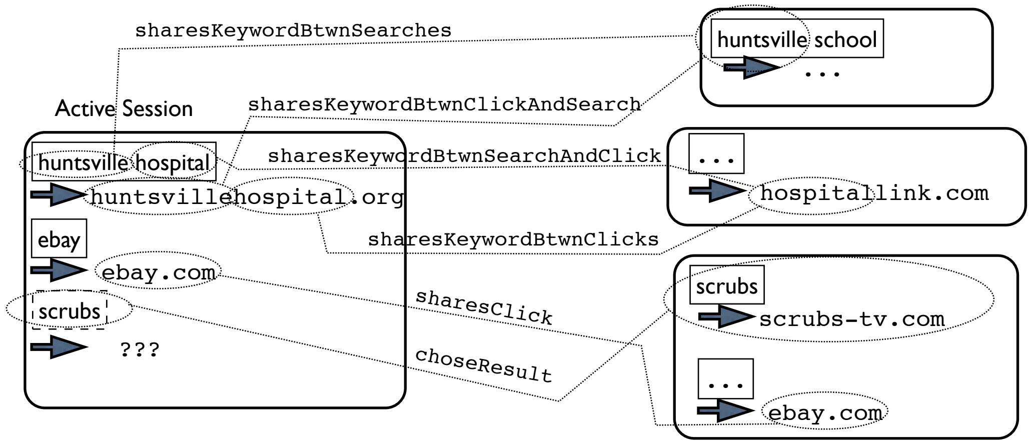


Figure 5.1: An illustration of predicates that relate sessions. Tokens in boxes represent queries, whereas tokens preceded by an arrow represent the clicked result for the preceding query. The *active* session, on the left, is related to some of the *background* sessions, on the right, by shared clicks or keywords. The tokens that are shared in each case are circled. Not all possible relations are drawn in order to reduce clutter.

choices made by related users from background sessions. For example, formula 1 establishes a relationship between the event that the active user chooses result R and the event that the user in a previous session S, related to the active session by sharing a click to a URL with hostname D, chose result R after searching for the current ambiguous query. Thus this formula exploits one type of relation between the active session and background sessions to provide evidence of the active user's intentions. This formula is always false when one of the first three evidence predicates is false, and in such cases it does not influence the probability that the active user chooses a particular search result. Thus, this formula plays a role only for background sessions that share clicks with the active session and chose a particular result R. The larger the number of such sessions, the stronger the belief that the

1:	$\texttt{Result}(\texttt{R}) \wedge \texttt{SharesClick}(\texttt{S}, \texttt{D})$ $\wedge \texttt{ChoseResult}(\texttt{S}, \texttt{R}) \wedge \texttt{ClickOn}(\texttt{R})$
2:	$\texttt{Result}(\texttt{R}) \wedge \texttt{SharesKeywordBtwnClicks}(\texttt{S}, \texttt{K})$ $\wedge \texttt{ChoseResult}(\texttt{S}, \texttt{R}) \wedge \texttt{ClickOn}(\texttt{R})$
3:	$\texttt{Result}(\texttt{R}) \wedge \texttt{SharesKeywordBtwnClickAndSearch}(\texttt{S}, \texttt{K})$ $\wedge \texttt{ChoseResult}(\texttt{S}, \texttt{R}) \wedge \texttt{ClickOn}(\texttt{R})$
4:	$\texttt{Result}(\texttt{R}) \wedge \texttt{SharesKeywordBtwnSearchAndClick}(\texttt{S}, \texttt{K})$ $\wedge \texttt{ChoseResult}(\texttt{S}, \texttt{R}) \wedge \texttt{ClickOn}(\texttt{R})$
5:	$\texttt{Result}(\texttt{R}) \wedge \texttt{SharesKeywordBtwnSearches}(\texttt{S}, \texttt{K})$ $\wedge \texttt{ChoseResult}(\texttt{S}, \texttt{R}) \wedge \texttt{ClickOn}(\texttt{R})$
6:	$\texttt{Result}(\texttt{R}) \wedge \texttt{ChoseResult}(\texttt{S}, \texttt{R}) \wedge \texttt{ClickOn}(\texttt{R})$
7:	$\texttt{Result}(\texttt{R}) \wedge \texttt{ClicksShareKeyword}(\texttt{R}, \texttt{D}, \texttt{K}) \wedge \texttt{ClickOn}(\texttt{R})$
8:	$\texttt{Result}(\texttt{R}) \wedge \texttt{ClickAndSearchShareKeyword}(\texttt{R}, \texttt{S}, \texttt{K}) \wedge \texttt{ClickOn}(\texttt{R})$
9:	$\texttt{Result}(\texttt{R1}) \wedge \texttt{Result}(\texttt{R2}) \wedge \texttt{R1} \neq \texttt{R2}$ $\wedge \texttt{ClickOn}(\texttt{R1}) \Rightarrow \neg \texttt{ClickOn}(\texttt{R2})$

Table 5.2: Formulae included in the model.

active user will also pick R; alternatively, the larger the number of such sessions, the greater the penalty for not picking R in the active session.

Formulae 2-5 encode analogous dependencies using each of the remaining session-relating predicates.[3]

Popularity Formula: Formula 6 in Table 5.2 encodes the intuition that the user will click the result that was the most popular among background users that searched for this ambiguous query. As before, the result for which there are the largest number of clicks in background data, and thus the largest number of groundings of this formula that are not falsified by the evidence, will have the largest prob-

[3]Although it may seem more natural to write these formulae as implications, i.e. $\texttt{Result}(\texttt{R}) \wedge \texttt{SharesClick}(\texttt{S}, \texttt{D}) \wedge \texttt{ChoseResult}(\texttt{S}, \texttt{R}) \Rightarrow \texttt{ClickOn}(\texttt{R})$, we found that defining the structure in this way leads to instability during weight learning.

ability of being clicked.

Local Formulae: Formulae 7-8 in Table 5.2 use information local to the active session to predict the user's preferences. Formula 7 (8) states that the user will click a result that shares keywords with a previous result (search) from the active session. We clarify that keywords were *not* extracted from the pages to which a URL points, but only from the URL itself because we are interested in developing a light-weight re-ranker. Because in our setting sessions are very short, we do not expect the local formulae to contribute much to the overall model performance. We include them in order to verify this.

Balance Formula: Finally, formula 9 in Table 5.2 sets up a competition among the possible results by stating that if the user clicks one of the results, the user will not click another one. This formula prevents all possible results from obtaining a very high probability of being clicked. This makes the model more discriminating and allows the same set of weights to perform well even as the number of groundings of the other formulae varies widely from one active session to the next.

It is worth noting that all of these formulae encode "rules of thumb" and useful features, which we expect will hold in general, but may sometimes be violated, e.g., the balance formula is violated when a user clicks more than one result for a query. The ability of MLNs to combine such varied sources of information effectively and in a principled way is one of the main considerations that motivated our choice of model.

Using these formulae, we defined three MLNs:

MLN 1 – Purely Collaborative: Contains only the collaborative formulae (1-5) and the balance formula (9).

MLN 2 – Collaborative and Popularity: Contains formulae 1-6 and the balance formula (9).

MLN 3 – Collaborative, Popularity, and Local: Contains all formulae. It can thus be viewed as a mixed collaborative-content-based model (e.g., Popescul et al., 2001; Melville et al., 2002).

5.3.2 Weight learning

To learn weights for the structures defined above, we used the contrastive divergence algorithm (CD) described by Lowd and Domingos (2007). CD can be viewed as a voted-perceptron-like gradient descent algorithm in which the gradient for updating the weight of formula C_i is computed as the difference between the number of true groundings of C_i in the data and the expected number of true groundings of C_i, where the expectation is computed by carrying out a small number of MCMC steps over the model using the currently learned weights. Like Lowd and Domingos (2007), we computed the expectations with MC-SAT (Poon & Domingos, 2006). We used the implementations of these algorithms in the Alchemy package (Kok et al., 2005), except that we adapted the existing implementation of CD so that learning can proceed in an online fashion, considering examples of sessions containing ambiguous queries one by one. This was done because otherwise our data was too large to fit in memory. We set the learning rate to

0.001 and the initial weight of formulae to 0.1 and kept all other parameters at their default values. Parameter values were selected on a validation set, strictly disjoint from our test set.

5.4 Data and Methodology

We used data provided by Microsoft Research containing anonymized query-log records collected from MSN Search in May 2006. The data consists of timestamped records for individual short sessions, the queries issued in them, the URLs clicked for each query, the number of results available for each query and the position of each result in the ranked results. We removed queries for which nothing was clicked. The average number of clicked results per session, over all sessions in the data, is 3.28. The data does not specify what criteria were used to organize a set of user interactions into a session; e.g., we do not know how multiple open tabs in a browser were treated. Although some of the sessions may belong to the same users, the data excludes this information through the lack of user-specific identifiers. This dataset therefore perfectly mirrors the scenario of disambiguating user intent from short interactions that we address in this research. Because there is a one-to-one correspondence between users and sessions, we will use these two terms interchangeably.

The data has two main limitations. First, it does not state which search queries are ambiguous. Automatically detecting ambiguity from user behavior is an interesting research question (e.g., Teevan, Dumais, & Liebling, 2008) but is not the focus of this work. We therefore employed a simple heuristic to obtain a (possibly noisy) set of ambiguous queries, using DMOZ (`www.dmoz.org`): a query string is considered ambiguous if, over all URLs clicked after searching for this exact string, at least two fall in different top-level categories, according to the DMOZ hierarchy. This heuristic does not require any human effort beyond that already invested in constructing DMOZ. Unfortunately, we could not include DMOZ category information into our models because many Web pages are not classified in the hierarchy. We limited ourselves to strings containing up to two words, thus obtaining $6,360$ distinct ambiguous query strings. Limiting the length of potentially ambiguous queries to two was motivated by the fact that most ambiguity occurs in short queries. For example, Sanderson (2008) found that the average length of ambiguous queries in two search log datasets ranges from 1.02 to 1.26 words. Queries of length at most two constituted 43.7% of all queries in our data. Of these queries, using the above method, we identified 2.4% as ambiguous, which agrees with the statistics reported by Sanderson, who found that between 0.8% and 3.9% of all queries are ambiguous (Sanderson, 2008). [4]

Another limitation of our data is that it does not list all the URLs presented to the user after a search but just the ones on which the user actually clicked. During testing, this is a problem because we do not know what possibilities to present to the system. To overcome this, we assumed that the set of all URLs clicked after searching for a particular ambiguous query string, over the entire dataset, was the set of results presented to the user. Our approach contrasts with that used in

[4]In Section 5.1, we cited Sanderson's findings for *frequently occurring* queries, whereas here we refer to his findings over *all* queries.

previous work, e.g., that of Dou et al. (2007), in which missing possible results lists are generated by separately querying the MSN search engine (on which data was collected) for each query. Although the queries were performed less than a month after the data was collected, the authors found that 676 queries from $4,639$ "lost the clicked web pages in downloaded search results." Because in our case almost 3 years have passed since the MSN06 data was collected, we preferred the simpler approach based on the available data. With this method, the average number of possible results for an ambiguous query string was 9.10. Figure 5.2 shows the distribution over the number of ambiguous queries for which we have a particular number of possible results. Although this heuristic is imperfect, it is likely to bias the results *against* our proposed solution—since every possible result was found to be relevant by at least one user, our systems cannot get high scores by simply separating the useful results from the totally irrelevant ones.

Figure 5.3 shows the distribution over the number of clicks preceding an ambiguous query in our test data. As can be seen, our test sessions, are indeed very short.

Several of the predicates we define use keywords. We next describe how we generated a list of keywords and how we extracted keywords from hostnames. To generate a list of keywords, we performed a pass over all training sessions. Any token separated by spaces was considered a keyword. As mentioned in Section 5.3, we then kept keywords that appeared at least 100 times and at most $10,000$ times. To determine which keywords occur in a given hostname, we first use the non-alphanumeric characters in the hostname to break it down into pieces and then

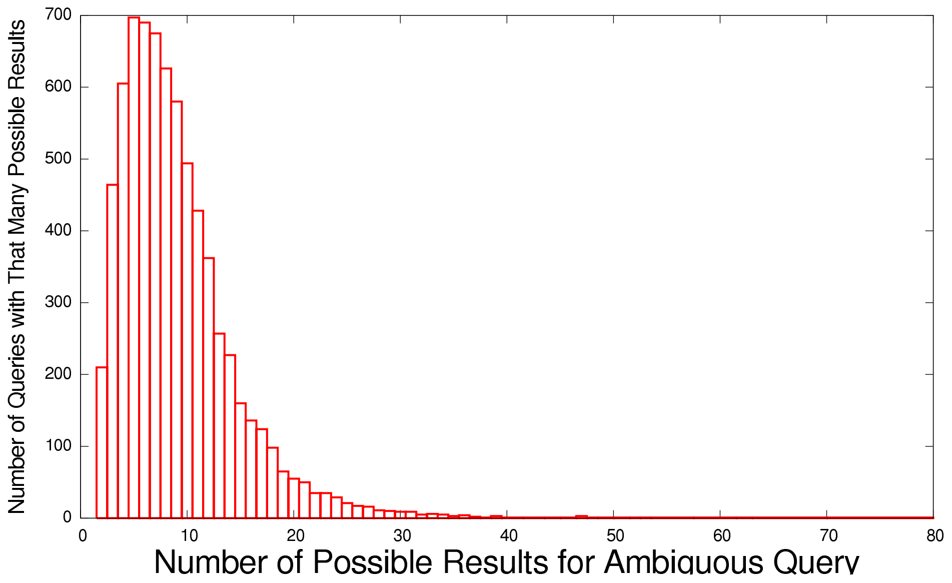


Figure 5.2: Histogram showing the distribution over the number of possible results available for an ambiguous query.

match each piece with keywords such that as much of the piece is covered as possible, using the smallest number of keywords.

To ensure a fair evaluation, the data was split into a training period and a testing period. The training period was used for training, validation, keyword generation, and idf (Manning, Raghavan, & Schutze, 2008) calculations (idfs were used by one of the baselines) and consisted of the first 25 days of data. The remaining 6 days of data were reserved for testing. Sessions that started in the training period and ended in the test period were discarded to avoid contaminating the test data. As validation/testing examples we used sessions that contained an ambiguous query from the training/testing periods respectively. To decrease the amount of random noise in the results, we removed from the test set sessions that contained

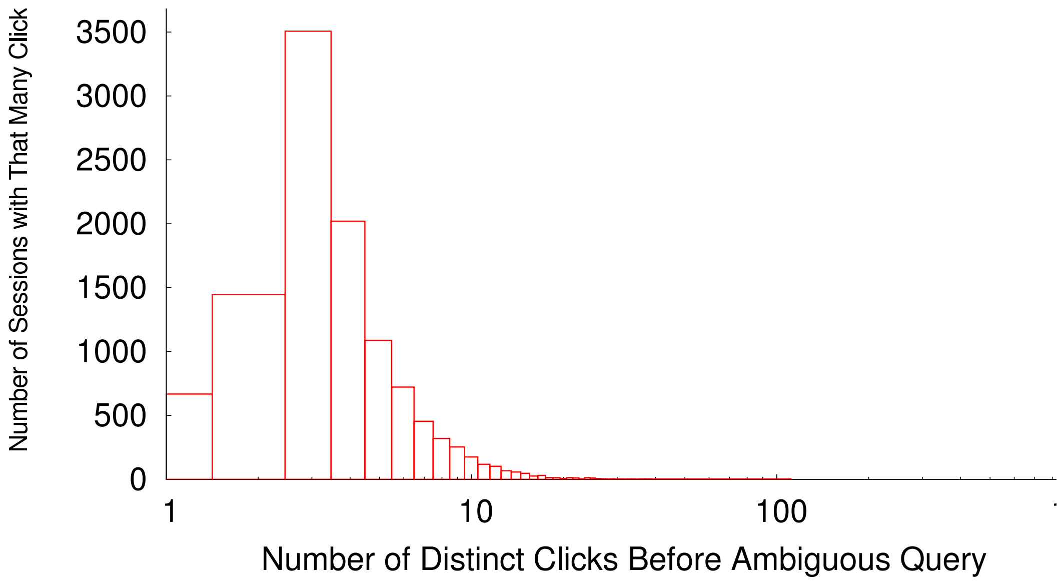


Figure 5.3: Histogram showing the distribution over the number of clicks preceding an ambiguous query in the test data. The X axis is drawn in log-scale.

no relational evidence, i.e., we removed the sessions that contain no true groundings of the `sharesKeyword/Click` predicates introduced in Section 5.3. In this way we obtained $11,234$ test sessions, which constitutes 72% of the available test sessions. The distribution over the number of previous clicks in these sessions is shown in Figure 5.3. As can be seen, the peak is at 3 distinct clicks before the ambiguous query.

During testing, only the information *preceding* the ambiguous query in the active test session is provided as evidence. The set of possible results for this ambiguous query string is given, and the goal is to rank these results based on how likely it is that they represent the intent of the user. The user may click more than one result after searching for a string. This behavior might be indicative of at least two possible scenarios: either the user is performing an exploratory search and all clicked results were relevant, or the user was dissatisfied with the results and kept clicking until finding a useful one. Since the data does not indicate which of these scenarios was the case, we treated all results clicked by the user after searching for the ambiguous query as relevant to his or her intentions. This presents yet another source of noise, and in the future we plan to explore approaches similar to the implicit feedback techniques described by Radlinski and Joachims (2005) to disentangle these possibilities, although the exact method introduced by these authors would not be applicable to our data because it requires the availability of an *ordered* list of the results returned to the user by the search engine. Another possibility is to use the time spent on a given page as an indicator of its relevance. User studies (e.g., Fox, Karnawat, Mydland, Dumais, & White, 2005) have confirmed the intuition that pages on which the user spends more time are more relevant to her search. Our data contains time-stamped records of user activities, so it is possible to obtain information on the amount of time spent on each clicked page except the last one within a session. We leave the exploration of this issue to future work.

Learning was performed as described in Section 5.3.2. To evaluate the learned models, we used Alchemy's implementation (Kok et al., 2005) of the MC-SAT algorithm (Poon & Domingos, 2006) for inference. During inference, we ran for 1,000 burn-in steps and 10,000 sampling steps. All other inference parameters were kept at their Alchemy defaults.

5.4.1 Evaluation Metrics

For evaluation purposes, the task of query disambiguation can be viewed as an information retrieval problem: rank the set of possible results so that the URLs reflecting the user's intentions (i.e., actually clicked by the user) appear as close to the top as possible. Thus, we used standard information retrieval metrics to evaluate the performance of our system (Manning et al., 2008) (Chapter 8):

(MAP) Area under the (interpolated) precision-recall curve, which is identical to the Mean Average Precision metric, commonly used by the IR community.

The MAP score is computed over a set of test instances T as follows:

$$\text{MAP}(T) = \frac{1}{|T|} \sum_{t \in \mathrm{T}} \frac{1}{|R_t|} \sum_{r \in R_t} \text{P@}r,$$

where R_t is the set of possible results for the t-th test instance and P@r is the precision of the top r results:

$$\text{P@}r = \frac{\text{Num relevant docs among the top } r}{r}.$$

(AUC-ROC) Area under the ROC Curve, which can be viewed as representing the mean average true negative rate. Using the notation from above, this metric is computed as follows:

$$\text{AUC-ROC}(T) = \frac{1}{|T|} \sum_{t \in \mathrm{T}} \frac{1}{|R_t|} \sum_{r \in R_t} \text{TN@}r,$$

where TN@r is the true negative rate of the top r results, defined as

$$\text{TN@}r = \frac{\text{Num irrelevant docs in positions } > r}{\text{Total num irrelevant docs}}.$$

Intuitively, the MAP measures how close the relevant URLs are to the top. One disadvantage of this metric in our case is that it is insensitive to the number of results to be ranked. For example, ranking a relevant result in the second position obtains the same score both when the number of possibilities is 2 and when it is 100, even though in the second case the task is clearly more difficult.

Assuming that the user starts scanning the page of returned results from top to bottom and does not consider any results appearing after the relevant ones, the AUC-ROC intuitively represents the percentage of irrelevant results that were *not* seen by the user before clicking on a search result. Thus, a random ranker would obtain an AUC-ROC of 0.5. Another useful characteristic of this measure is that unlike the MAP, it is sensitive to the number of possible results that are to be ranked.

A final issue is how to break ties when a relevant result has the same score as some irrelevant results. We report the **average case** in which the relevant result is placed in the middle position within the group of results with equal scores. For the most interesting systems, we also report the **worst case** in which the relevant result is placed last within the group of results that share scores. This is motivated by the goal of performing effective personalization *consistently*. The best case is not interesting because for it perfect performance can be obtained by giving all results the same score.

5.4.2 Systems Compared

We compared the MLNs from Sect. 5.3 to several baselines:

Random: Ranks the possible results randomly.

Collaborative-Pearson: Implements a standard collaborative filtering algorithm (Herlocker et al., 1999) that weights each previous user based on the Pearson correlation between the preferences (i.e. clicks) of that user and the active user. We considered a clicked result to have rating 1, and an unclicked result that was clicked by another user for the same query to have rating 0, and all other results to be unrated. The n closest neighbors are chosen (we used $n = 30$ following (Herlocker et al., 1999)), and the prediction that a given result is selected is formed as a weighted average of the deviations from the mean of each neighbor.

Collaborative-Cosine: Identical to **Collaborative-Pearson** except that it computes the similarity between the active user and a previous user as the cosine similarity between the idf-weighted vectors of their clicked results.

Popularity: Ranks each result according to the number of previous sessions that searched for the ambiguous query and chose it.

The goal of query disambiguation is to improve the result ranking over that obtained by just using the general ranker. Thus, a natural baseline is the general ranker itself, which in our case is the MSN search engine. Because the position in the ranked list of each clicked result is available from our data, we could compute MAP and AUC-ROC scores for the MSN search engine based on these positions. However, because people have a strong bias towards clicking the top result on a page (Joachims, Granka, Pan, Hembrooke, & Gay, 2005), such a comparison would give an unfair advantage to the MSN search engine. Moreover, the set of results that are displayed and the ranking of those results tend to shift frequently (Teevan et al., 2008), thus a highly relevant result may not have been clicked simply because it did not appear in the list displayed to the user. Finally, such a baseline does not take into account that there may be at least two results that satisfy an information need equally well. Thus, a fairer comparison to the search engine requires actually deploying our proposed systems and testing their effectiveness with real users. Unfortunately, we do not have the resources necessary to launch such a study.

5.5 Results

Table 5.3 presents the performance when ties among results with the same score are broken as in the average case. The **Collaborative-Pearson** baseline performs no better than **Random** on AUC-ROC and only slightly better than **Random** on MAP. Switching to cosine similarity in **Collaborative-Cosine** gives modest (but significant) improvements. The **Popularity** baseline is very strong and outperforms the other baselines, as well as **MLN 1**. However, combining popularity with relational information in **MLN 2** leads to significant gains in performance, and **MLN 2** achieves a significantly higher AUC-ROC score. **MLN 2**, our strongest model, highlights the main advantage of using MLNs: we were able to significantly improve **MLN 1** by incorporating a reliable source of information simply by adding the popularity formula to the model. Finally, as expected, we observe that adding local formulae in **MLN 3** does not improve performance. This demonstrates that the interactions of the active user prior to the ambiguous query are not directly helpful for determining intent and occurs as a result of the brevity of sessions in our data (cf. Figure 5.3). The inefficacy of local formulae may also be due to the fact

System	**MAP**	**AUC-ROC**
Random	0.317	0.502
Collaborative-Pearson	0.333	0.502
Collaborative-Cosine	**0.360**	**0.521**
Popularity	**0.389**	**0.575**
MLN 1	0.375	0.563
MLN 2	0.386	**0.587**
MLN 3	0.366	0.583

Table 5.3: Results over all test sessions that contain an ambiguous query when ties in ranking are broken as in the average case. Numbers in bold present significant improvements over all preceding systems at the 99.996% confidence level according to a paired t-test. Additional significant differences are: **MLN 1** is a significant improvement over all baselines except **Popularity**, and **MLN 2** improves significantly over all preceding systems except for **Popularity** also in terms of MAP; there is no significant difference between the MAP scores of **Popularity** and **MLN 2**;the MAP score of **Popularity** is significantly higher than that of **MLN 1.**

that a session may continue when the user is dissatisfied with the results obtained so far. It is interesting to contrast this result with the findings of Dou et al. (2007) who experimented with much longer sessions (up to 12 days) and reported that the previous interactions of the active user presented a very strong signal for personalization purposes. This emphasizes a fundamental difference in the assumptions on the data made in this versus previous research: because in our case user-specific session information is so limited, we cannot rely on only using the past preferences of the active user and must instead exploit relations to other, historical, users.

Next, we analyze in more detail the performance of the MLN systems to that of **Popularity**, which is the strongest baseline. Table 5.4 presents the performance over all test sessions when ties in ranking are broken as in the worst case. As can be

System	MAP	AUC-ROC
Popularity	0.380	0.525
MLN 1	0.373	**0.563**
MLN 2	**0.385**	**0.586**
MLN 3	0.355	0.572

Table 5.4: Results over all test sessions that contain an ambiguous query when ties in ranking are broken as in the **worst case.** Numbers in bold present significant improvements over all preceding systems at the 99.996% confidence level according to a paired t-test. Additionally, the MAP score of **Popularity** is significantly higher than that of **MLN 1.**

seen, **Popularity**'s AUC-ROC score decreases sharply, whereas the MLN models maintain their performance to almost the same level as in the average case. This behavior is observed partly because **Popularity** introduces many more ties among the scores of possible results than do the MLN models. In particular, averaged over all test sessions, the ratio between the number of possible results and the number of distinct scores for **Popularity** was 1.8, whereas for **MLN2** it was just 1.02. These results indicate that **Popularity**'s behavior is erratic and can, for the same user and the same query, lead to rankings that vary highly in quality. This kind of behavior can give the perception of poor quality to a frequent user. On the other hand, the MLN models are consistent, maintaining the quality of their rankings in the worst case.

Finally, we compare the performance of **Popularity** to that of **MLN 2** while varying the degree to which some of the possible results for an ambiguous query dominate in popularity over the rest. We formalized this as follows. Let q_Q be the empirical distribution over the results clicked for an ambiguous query Q. This distribution was measured empirically on the training data, i.e., for every ambiguous query, we determined from the training sessions the proportion of time each potential search result was clicked. We then separated the test examples into bins, such that bin i contains all test sessions s for which $\lfloor KL_{q_Q||\text{uniform}} \rfloor = i$, where Q is the ambiguous query in session s and $KL_{q_Q||\text{uniform}}$ is the KL divergence of q_Q to the uniform distribution. In other words, bin 0 contains the sessions in which the possible results for the ambiguous query were all chosen with roughly the same frequency. Higher-numbered bins contain sessions in which one of the search results strongly dominates in popularity over the other possibilities. When this is the case, predicting just based on the popularity of a result gives good performance. The more challenging scenario occurs in the lower-numbered bins where the preferences over possible results are more uniformly distributed. Figures 5.4 and 5.5 compare **Popularity** to **MLN 2** when ties in ranking are broken for the average and worst cases respectively. **MLN 2** maintains a lead over **Popularity** until the last two bins in which the distribution over possible results is furthest from uniform. As we expect, the difference between the performance of the two systems shrinks as we move to higher-numbered bins, and **MLN 2** has a greater advantage over **Popularity** in the lower-numbered bins in which the need to disambiguate is more pressing. The sharp drop in accuracy observed in bin 7 is due to the fact that one of the ambiguous queries occurring in sessions in this bin was overwhelmingly followed by clicks to what seems to be a newly appearing Web page during the test period. That page was selected only 3 times in the training period while the most popular page in the training period was selected more than 2000 times.

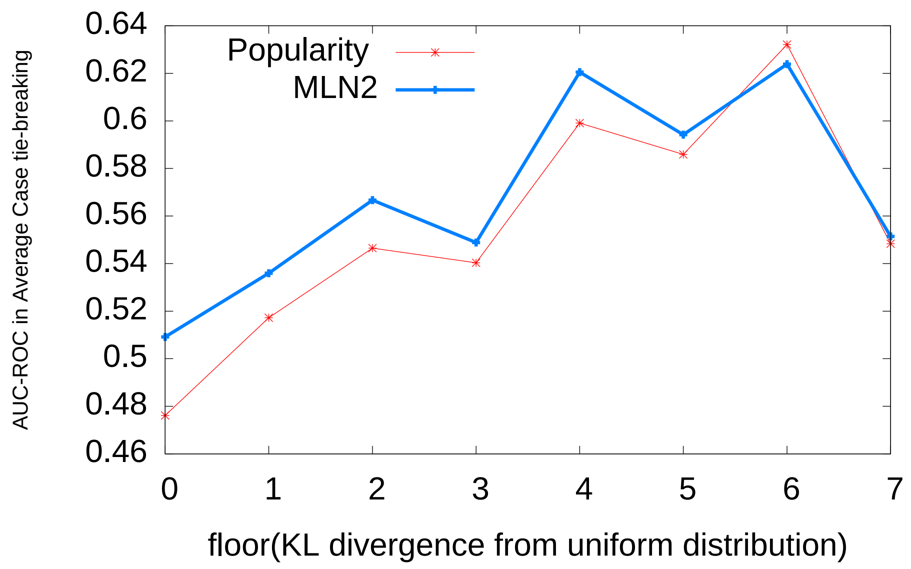


Figure 5.4: AUC-ROC when ranking ties are broken so as to simulate the **average case** for different bins of KL divergence of the distribution over possible results to uniform.

As a final but important note, inference over the learned models was very efficient and completed in the order of a second.

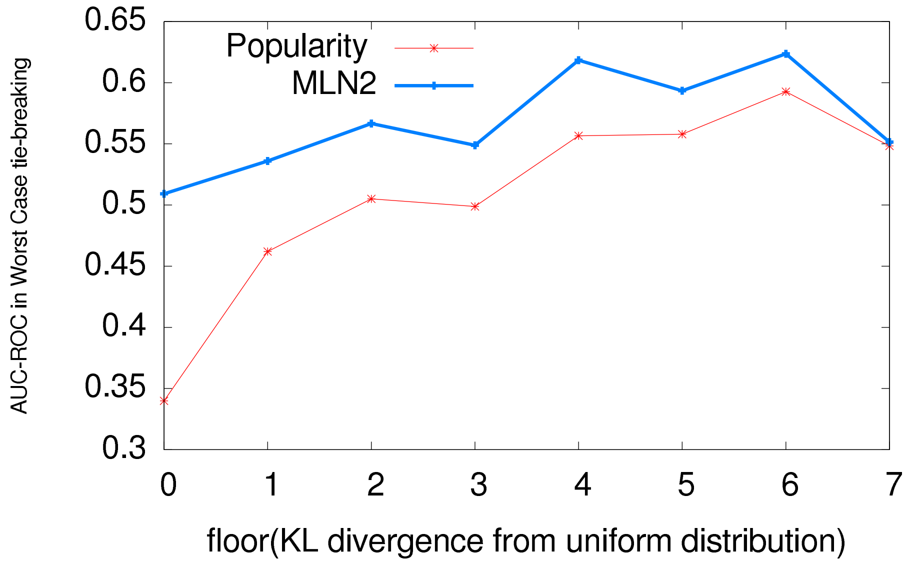


Figure 5.5: AUC-ROC when ranking ties are broken so as to simulate the **worst case** for different bins of KL divergence of the distribution over possible results to uniform.

Chapter 6

Future Work

This chapter describes some ways in which the contributions of this thesis can be extended. We consider future work relating to transfer learning, structure learning, and applications to Web disambiguation.

6.1 Transfer Learning

As we discussed in Section 3.1, transfer learning has been applied to a wide range of problems and settings. The strong interest in this area is motivated not only by the intellectual appeal of transfer learning as an approach that better emulates the way humans learn, but also by the fact that transfer learning techniques have proven effective in addressing many challenging problems. We envision several ways in which our contributions to transfer learning can be extended.

6.1.1 Integrating Mapping and Revision

TAMAR views mapping and revision as two separate and independent aspects of transfer across multi-relational domains. An interesting extension would be a system that instead integrates these two processes. The advantages of such a system are that it would provide both a way of gauging the usefulness of source knowledge, and a more efficient mapping procedure. Such a system could operate as follows, supposing that the task is to learn MLN structure.

Acquiring preliminary knowledge: Given a new target domain, learning starts from scratch, focusing on acquiring what we will call *preliminary* knowledge that is easier and faster to extract from data than is a full model. Such knowledge could consist of short clauses that capture dependencies between pairs of relations or a data structure such as the Markov network template from Chapter 4.

Preliminary knowledge guides mapping: Preliminary knowledge can be helpful in guiding the mapping process. For example, both TAMAR and SR2LR consider every possible type-consistent predicate mapping. While this process is extremely efficient in our domains, it could become prohibitively expensive in domains with a large number of predicates that all take the same types of arguments. Thus, an algorithm that uses the preliminary knowledge to guide predicate mapping would be more effective in the latter situations. Such an algorithm can start by establishing structural correspondences between the source model and the preliminary target knowledge, akin to how it is done in the structure-mapping engine (Falkenhainer et al., 1989) (described on page 38). If the preliminary knowledge consists of short clauses, structural correspondences will be established only with the short clauses in the source model. If instead the preliminary knowledge is represented as a Markov-network-template-like data structure, structural correspondences will be established between the vliteral dependencies implied by the source clauses and those captured in that data structure. Because the preliminary knowledge does not represent all aspects of the target model, in a process analogous to that in SR2LR, the predicate mappings implied by these correspondences can be used to transfer additional source clauses that would require more effort to learn from scratch.

Preliminary knowledge as a relatedness gauge: Preliminary knowledge can also serve as a basis for relatedness measures that estimate the similarity between two domains. For example, if large portions of the preliminary knowledge cannot be mapped to the source model, this can be taken as indication that the source and target domains are not sufficiently close. Relatedness measures based on preliminary knowledge can also be used to perform source selection, allowing the transfer system to determine autonomously which from a set of source models is closest to the target domain. In fact, when models from several previously encountered domains are available, the system can perform transfer from multiple sources rather than limiting itself to a single source. This capability would be especially useful when no single previously learned model is a good match for the target domain. In such cases, a combination of two or more sources, each of which represents a different aspect of the target domain, could be effective.

Evaluation of Mapped Knowledge and Revision: Rather than evaluating possible mappings with a probabilistic measure, as done by MTAMAR, better results could be obtained by using all mappings of the source clauses that fit the structural correspondences with the preliminary knowledge, attempting to revise them, and dropping them only if they are ineffective even after the revision. This is motivated by the observation in Figure 3.10 that SR2LR can outperform MTAMAR even when knowledge about the domain grows. The revised structure can then be used to find better mappings of the source clauses, which could allow more of the source clauses to be transferred, thus alternating between mapping and revising the source knowledge.

6.1.2 Bottom-Up Revision

A second direction in which our work on transfer and structure learning can be extended is by developing a bottom-up learner, such as BUSL, that can be used not only for learning from scratch but also for revision of existing knowledge. Such a revision algorithm can be used to revise both transferred and human-provided knowledge. In preliminary experiments with such algorithms, we found that, given complete domain knowledge of at least one mega-example, BUSL obtained better predictive accuracy when learning from scratch than it did when revising transferred knowledge. However, we expect that if target-domain data is incomplete, revision algorithms that, like SR2LR, are aware of the missing data would lead to more accurate models. Such algorithms could operate analogously to SR2LR by revising only those aspects of existing knowledge that can be reliably evaluated on the available data and using insights from these revisions to also correct the remaining aspects.

A related problem is the need for systematic studies of how varying the number of unknown facts in a domain affects the relative performance of systems that learn from scratch or use transfer learning. Typically, in SRL applications one makes the closed-world assumption (CWA) as a convenient way of storing the data. Under the CWA only the true facts need to be stated, and any fact that is left out is assumed to be false. In Section 3.3, we considered one way in which the CWA can be modified by restricting it so that it applies to a single entity. An interesting future question is to study other ways in which the CWA can be qualified.

6.2 Structure Learning

We can view MLN structure as serving two distinct purposes. On the one hand, the logical formulae capture dependencies and regularities among the relations, such as that if someone teaches courses, then she also advises students. On the other hand, clauses can also serve as relational features that describe complex relational characteristics of the entities in the data. This happens frequently in molecular biology domains where the clauses are used to describe aspects of the chemical structure of the molecules, such as benzene rings. The TNode construction procedure of BUSL, described in Section 4.2.1 discovers relational features, whereas the Markov network template construction from Section 4.2.2 finds dependencies among these features. However, at present TNode construction is limited to finding relational features consisting of at most two literals. Although in principle the procedure could discover longer TNodes, the size of the search space explodes quickly as the TNode length grows. In many cases, it may be necessary to discover longer features; e.g., to describe a benzene ring, one needs to capture the relations among six carbon-hydrogen pairs. The LHL algorithm, recently developed by Kok and Domingos (2009), comes with an efficient procedure for discovering longer relational features. Thus, in the future, it would be interesting to explore ways in which LHL's approach can be used by BUSL to discover more descriptive TNodes, that could then be related to each other in the Markov network template construction step. This would allow for the efficient discovery of dependencies among complex relational characteristics.

6.3 Web Query Disambiguation

Our work on exploiting relational information to compensate for insufficient entity-specific data in Web query disambiguation motivates several avenues for future research.

Better disambiguation accuracy can be obtained by incorporating more evidence into our models. For example, our current approach relates the active session only to sessions that also searched for that ambiguous query. In addition, we envision including relations to sessions that did not search for that exact query but clicked on a possible result for it. Additional information can also be provided by bringing in outside sources, such as the actual content of possible results, or topic categories in which they participate.

One prerequisite to efficient modeling with such diverse sources of information is the ability to retrieve knowledge relevant to a new user efficiently. For example, one of the formulae we used in Chapter 5 was:

$$\mathtt{Result}(\mathtt{R}) \wedge \mathtt{SharesClick}(\mathtt{S}, \mathtt{D}) \wedge \mathtt{ChoseResult}(\mathtt{S}, \mathtt{R}) \wedge \mathtt{ClickOn}(\mathtt{R})$$

The `SharesClick` and `ChoseResult` predicates in this formula refer only to sessions that contain a search for the ambiguous query from the current session. This is a much smaller set than the set of all sessions that contain at least one click to a possible result for the ambiguous query. Thus, while in our existing model efficiency is guaranteed by the size of the population with which relations are established, if we increase this population in order to obtain richer evidence, efficient retrieval of relevant sessions becomes extremely important. Some work in this direction has already been done in the recent FROG system (Shavlik & Natarajan, 2009) and in the implementation of clause grounding in Alchemy (Kok et al., 2005). However, we believe that more efficient indexing schemes, closely coupled with SQL databases in which such data can be conveniently stored, would lead to dramatic improvements in efficiency.

A second direction of future work motivated by Web query disambiguation is learning of more nuanced models. Currently, our system learns a single weight for each formula. However, some shared domains (represented by the D variable in the formula above) are better predictors of relatedness than others. For instance, we expect that a shared click to `yahoo.com` is less indicative of relatedness than is a shared click to `ijcai.org`. In preliminary experiments, we attempted to learn a separate weight for each possible relating domain in each formula but found that the available training data was too sparse to support such an approach. A better technique would be to first cluster the domains according to their ability to relate sessions and learn a separate weight for each rule and each cluster. Approaches that cluster entities in multi-relational data have already been developed (e.g., Kok & Domingos, 2007, 2009). In this case, however, we expect that simpler techniques that can handle training data coming in as a stream rather than in a batch would work better because of the large size of the data.

Finally, at present our evaluation procedure is limited by the fact that our data does not list all results presented to a user but just the clicked ones. We would like to explore ways in which this process can be made less noisy, such as, for example, by taking into consideration the amount of time spent on a clicked result. It would also be interesting to test our system in action, i.e., as part of an experimental search engine on new ambiguous queries.

6.4 Other SRL Models

Because of the generality of MLNs, many of the ideas we have presented in this thesis can be applied to other SRL models. In Section 3.3.1, we already discussed other models that could be used to perform transfer learning with SR2LR. Similarly, the main idea used in BUSL can be employed to train other SRL models, in particular Bayesian logic programs (BLPs) (Kersting & De Raedt, 2001). A BLP defines a Bayesian network via a set of Horn clauses, each of which specifies a dependence of the head on the antecedents. This is analogous to MLNs in which first-order formulae define dependencies among their literals. Thus, BLP structure could be learned using a BUSL-like algorithm, that first discovers sets of interdependent variables, as in the Markov network template from Chapter 4, and then searches for Horn clauses that comply with these dependencies.

Chapter 7

Conclusions

The research presented in this thesis addresses several aspects of learning with Markov logic networks (MLNs). We have motivated and followed two main themes: the effectiveness of bottom-up learning techniques that use the available data not only to evaluate hypotheses but also to propose them; and the need for methods that allow for effective modeling from limited data. Adopting these themes, we have addressed the problems of structure learning from scratch, transfer learning, and Web query disambiguation.

We first focused on the problem of transfer learning across relational domains, addressing two different settings. In the first setting, a sufficient amount of target-domain data is available, and the goal is to revise a transferred structure so that it obtains better predictive accuracy in the target domain. We developed an algorithm that first diagnoses the source structure in order to determine which parts of it do not fit the target task. This diagnostic analysis then allows revision to focus only on the incorrect portions of the structure, thus speeding up learning in the target task. To find dependencies that are new to the target domain, our algorithm incorporates ideas from inductive logic programming and implements relational pathfinding, an effective procedure based on finding paths in the relational graph of the data.

In the second transfer learning setting we considered, target-domain data is severely limited, consisting of information about a handful of entities, in the extreme case just one. When such a limitation is placed on data, effective learning from scratch is infeasible and transfer learning is a natural approach. We developed a simple but effective technique that maps source knowledge to the target domain by testing out potential predicate mappings on short-range clauses whose correctness can be directly evaluated on the available data. Successful mappings are then used to map the remaining clauses. We demonstrated that in this way reasonable accuracy can be attained from very limited data, and that this approach is superior to several baselines, as well as to a technique that is not explicitly addressing the missing data aspect.

A second problem we addressed in this thesis is structure learning from scratch. This problem is important not only as a way to obtain source models for transfer but also for modeling in stand-alone tasks. We developed a bottom-up structure learner that starts by discovering a Markov network template, a novel data structure that encodes the dependencies among unground literals. The Markov network template then guides the search for clauses. In this way, our algorithm can avoid some of the pitfalls of top-down approaches, such as local maxima and plateaus.

In the final part of the thesis, we focused on a specific problem, that of Web query disambiguation, to demonstrate how by exploiting relations between entities, we can compensate for a constraint on the amount of entity-specific information that is available. We defined several ways of relating the sessions of search engine users and defined the structure of an MLN based on these relations. Weights for this structure were then learned from the data. We demonstrated that our approach outperforms several natural, and in some cases, strong baselines.

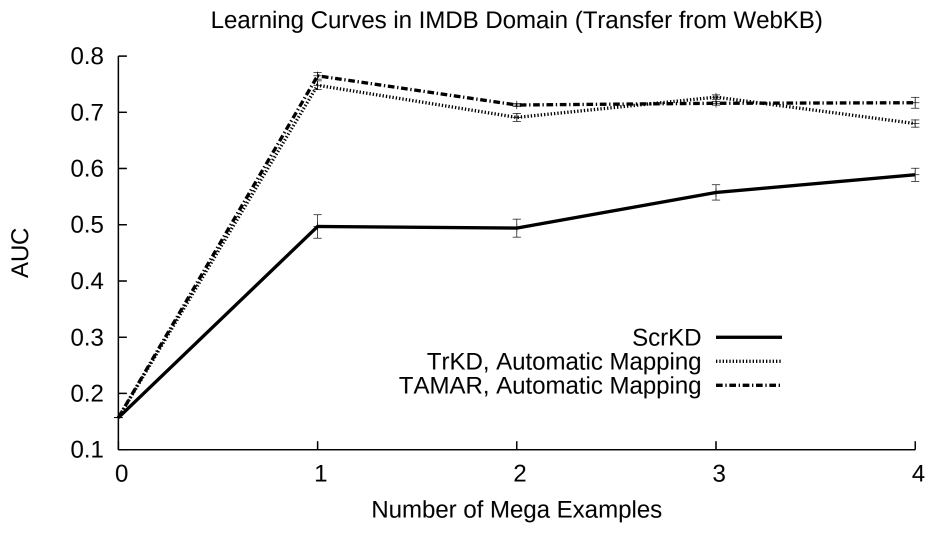
Overall, the contributions in this thesis have led to progress on structure learning, a core aspect of successful modeling in multi-relational domains, as well as to progress on a practically significant application of SRL to Web query disambiguation. We hope that our work will lead to wider use of bottom-up learning in the SRL community and to the introduction of SRL techniques to enable advances in new problems, such as ones in Web personalization, that have traditionally been viewed as feature-vector tasks.

Appendix

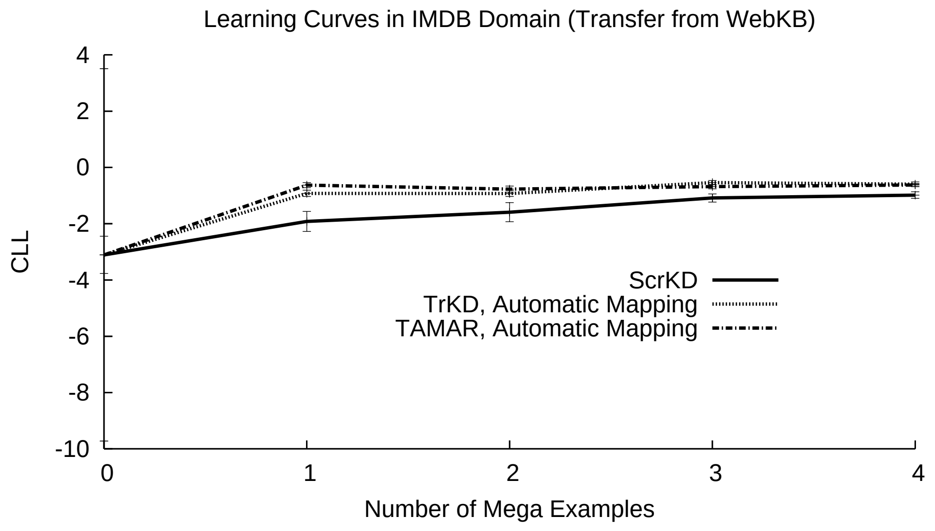
Appendix 1

Complete Learning Curves of TAMAR

Figures 1.1 to 1.5 present complete learning curves for the results presented in Section 3.2.2. The zeroth points are obtained by testing the performance of the MLN provided to the learner at the start.

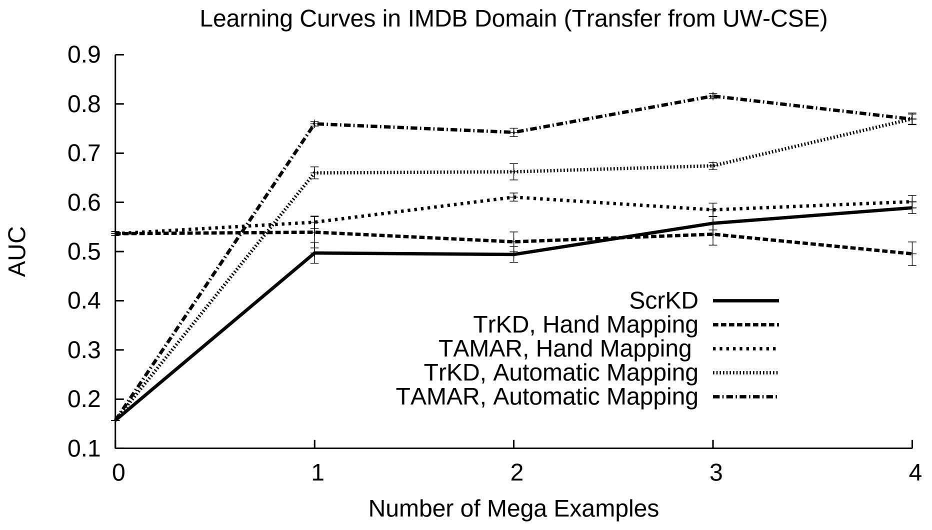


a)

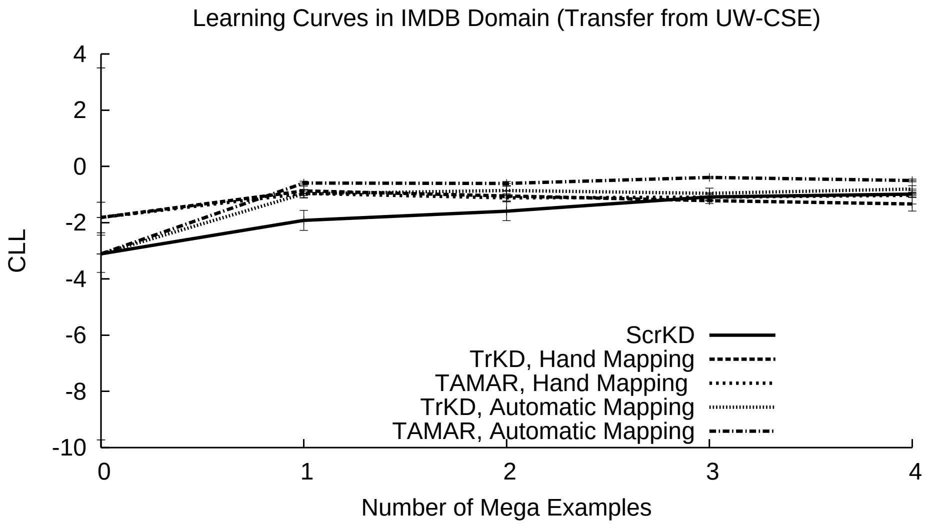


b)

Figure 1.1: Learning curves in WebKB → IMDB for a) AUC and b) CLL.

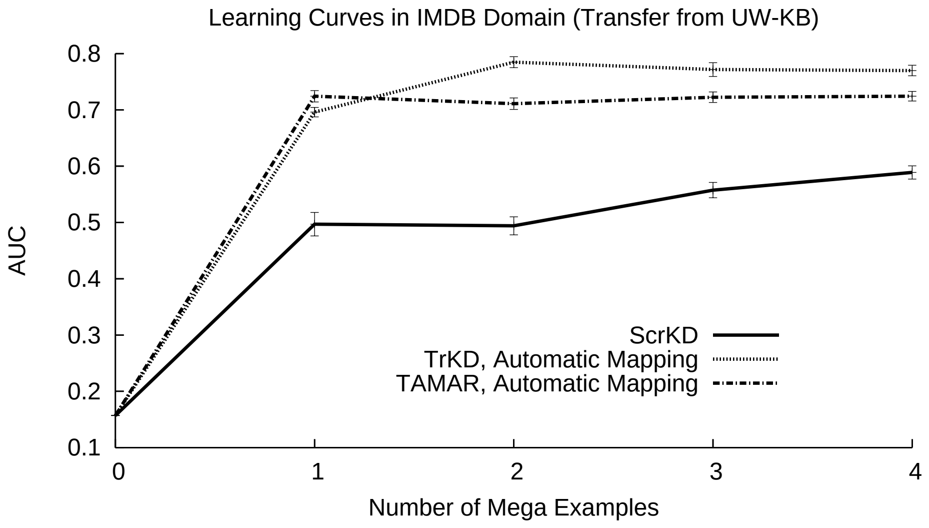


a)

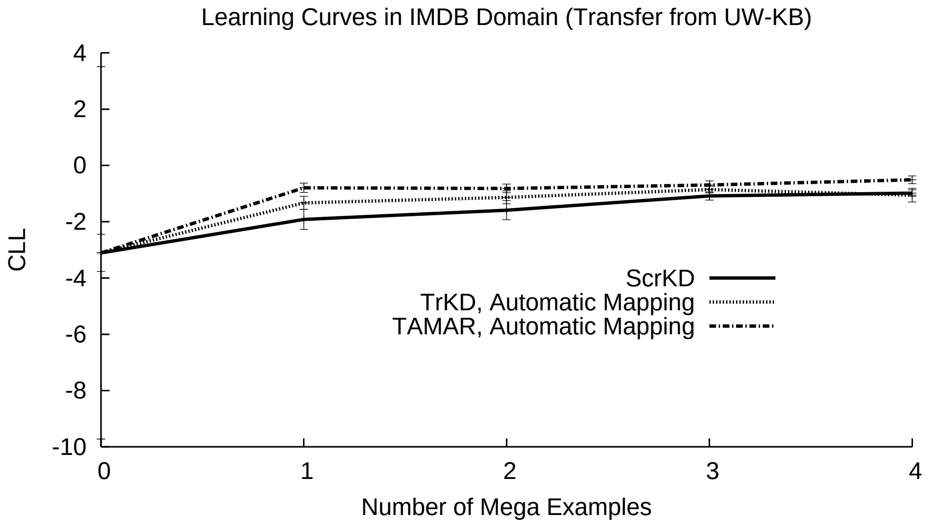


b)

Figure 1.2: Learning curves in UW-CSE → IMDB for a) AUC and b) CLL. Here we additionally tested the performance of systems that do not use the automatic mapping but are provided with an intuitive hand-constructed mapping that maps Student → Actor, Professor → Director, AdvisedBy/TempAdvisedBy → WorkedFor, Publication → MovieMember, Phase → Gender, and Position → Genre.

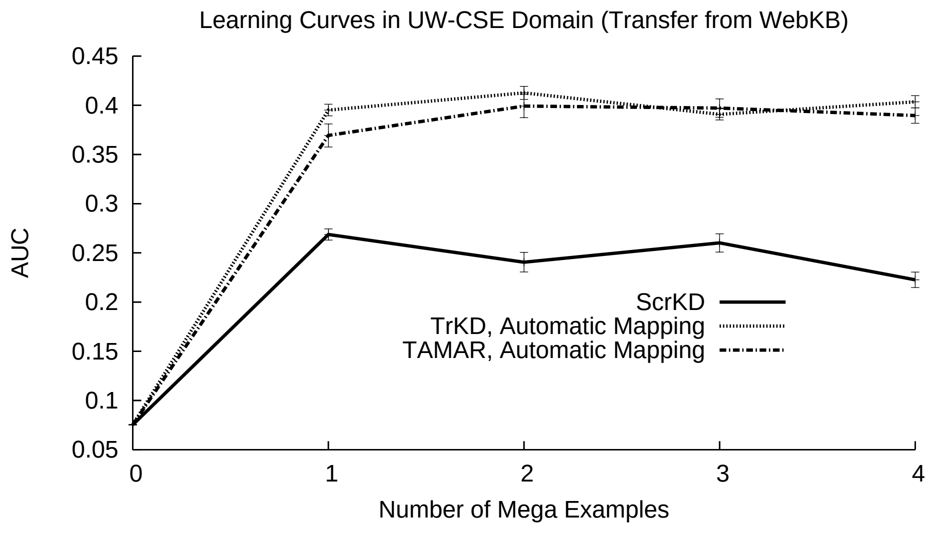


a)

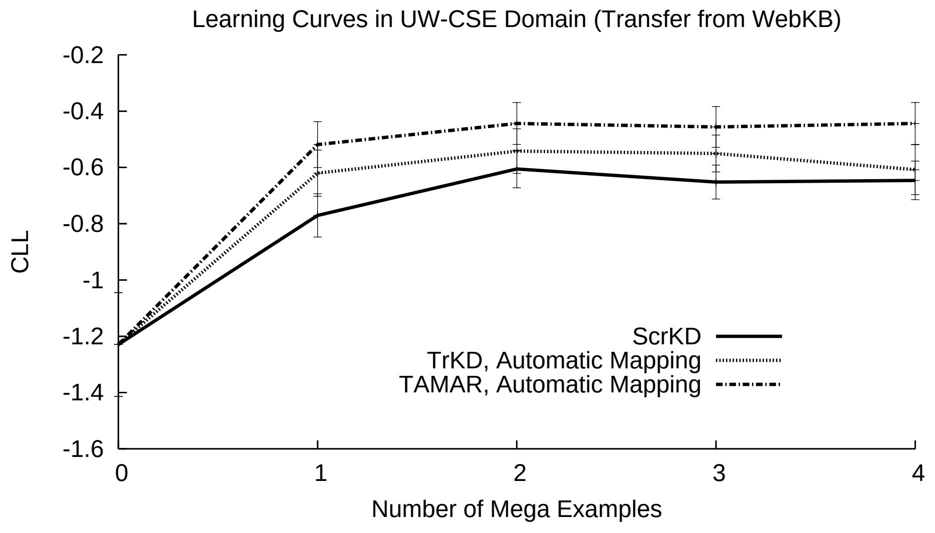


b)

Figure 1.3: Learning curves in UW-KB → IMDB for a) AUC and b) CLL.

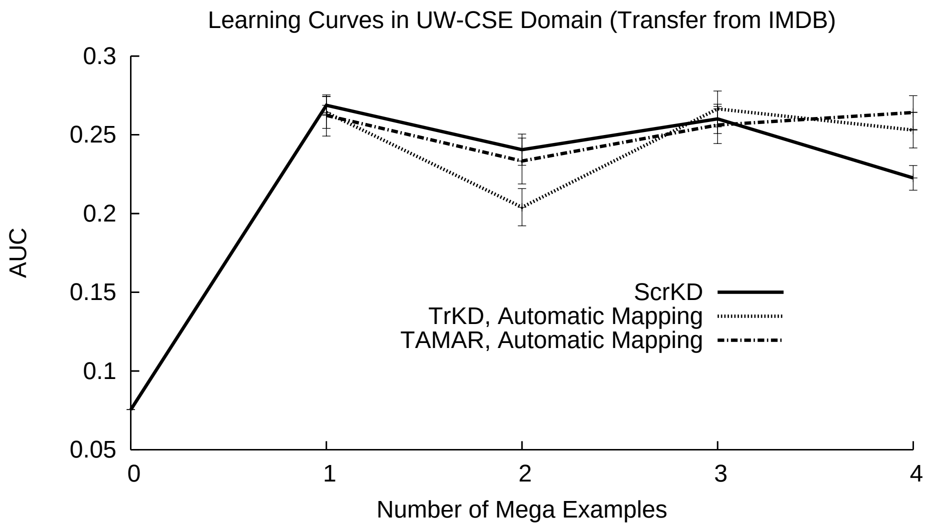


a)

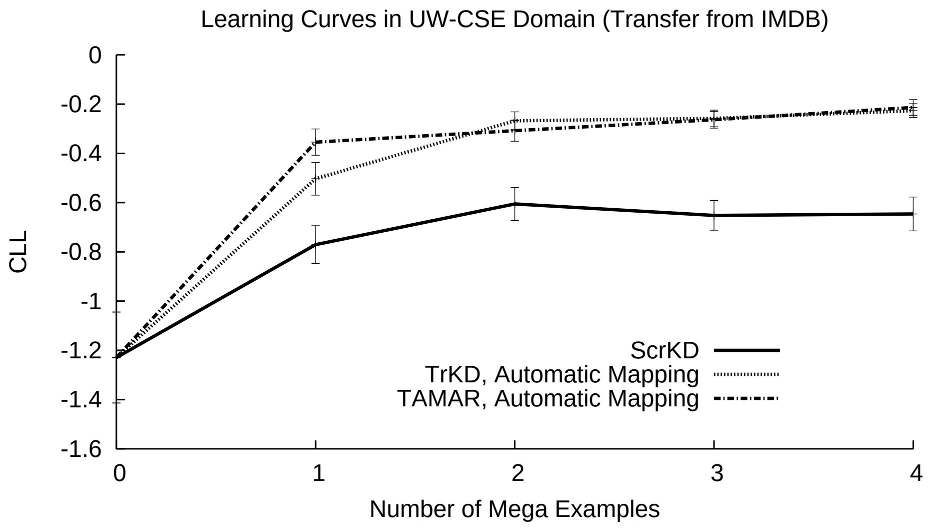


b)

Figure 1.4: Learning curves in WebKB $\rightarrow$ UW-CSE for a) AUC and b) CLL.



a)



b)

Figure 1.5: Learning curves in IMDB → UW-CSE for a) AUC and b) CLL.

References

Abbeel, P., Koller, D., & Ng, A. Y. (2006). Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research*, *7*, 1743–1788.

Agichtein, E., Brill, E., & Dumais, S. (2006). Improving Web search ranking by incorporating user behavior information. In *Proceedings of 29th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-06).* Seattle, WA.

Almeida, R. B., & Almeida, V. A. F. (2004). A community-aware search engine. In *Proceedings of the Thirteenth International World Wide Web Conference (WWW-04).* New York, NY.

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, *6*, 1817–1853.

Arias, M., Khardon, R., & Maloberti, J. (2007). Learning Horn expressions with LogAn-H. *Journal of Machine Learning Research*, *8*, 549–587.

Banerjee, B., Liu, Y., & Youngblood, G. M. (Eds.). (2006). *ICML workshop on "Structural Knowledge Transfer for Machine Learning".*

Banerjee, B., & Stone, P. (2007). General game learning using knowledge transfer. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07).* Hyderabad, India.

Barbaro, M., & Zeller, T. (2006). *A face is exposed for AOL searcher no. 4417749.* New York Times. (`http://www.nytimes.com/2006/08/09/technology/09aol.html?ex=1312776000`. Accessed on 16 Oct. 2008)

Berger, A. (1996). *A brief maxent tutorial.* (Available at `http://www.cs.cmu.edu/afs/cs/user/aberger/www/html/tutorial/tutorial.html`)

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B (Methodological)*, *48*(3), 259-302.

Biba, M., Ferilli, S., & Esposito, F. (2008). Discriminative structure learning of Markov logic networks. In *Proceedings of the 18th International Conference on Inductive Logic Programming (ILP-08).* Prague, Czech Republic.

Bilenko, M., & Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)* (pp. 39–48). Washington, DC.

Blitzer, J., Dredze, M., & Pereira, F. (2007). Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07).* Prague, Czech Republic.

Blitzer, J., McDonald, R., & Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-06).* Sydney, Australia.

Bonilla, E. V., Williams, C. K. I., Agakov, F. V., Cavazos, J., Thomson, J., & O'Boyle, M. F. P. (2006). Predictive search distributions. In *Proceedings of 23rd International Conference on Machine Learning (ICML-06).* Pittsburgh, PA.

Breese, J. S., Heckerman, D., & Kadie, C. (1998). *Empirical analysis of predictive algorithms for collaborative filtering* (Tech. Rep. No. MSR-TR-98-12). Microsoft Research.

Bromberg, F., Margaritis, D., & Honavar, V. (2006). Efficient Markov network structure discovery using independence tests. In *Proceedings of the Sixth SIAM International Conference on Data Mining (SDM-06).* Bethesda, MD.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., et al. (2005). Learning to rank using gradient descent. In *Proceedings of the 22th International Conference on Machine Learning (ICML-05).* Bonn, Germany.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Chen, H., & Karger, D. R. (2006). Less is more: Probabilistic models for retrieving fewer relevant documents. In *Proceedings of 29th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-06).* Seattle, WA.

Cohen, P. R., Chang, Y., & Morrison, C. T. (2007). Learning and transferring action schemas. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07).* Hyderabad, India.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02).* Philadelphia, PA.

Conti, G. (2006). Googling considered harmful. In *New Security Paradigms Workshop.*

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., et al. (1998). Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)* (pp. 509–516). Madison, WI.

Croonenborghs, T., Driessens, K., & Bruynooghe, M. (2007). Learning relational options for inductive transfer in relational reinforcement learning. In *Proceedings of the 17th International Conference on Inductive Logic Programming (ILP-07).* Corvallis, OR.

Davis, J., & Domingos, P. (2008). Deep transfer via second-order Markov logic. In *Proceedings of the AAAI workshop on Transfer Learning for Complex Tasks.* Chicago, IL.

Davis, J., & Domingos, P. (2009). Deep transfer via second-order Markov logic. In *Proceedings of the 26th International Conference on Machine Learning (ICML-09).* Montreal, Quebec.

Davis, J., Ong, I., Struyf, J., Costa, V. S., Burnside, E., & Page, D. (2007). Change of representation for statistical relational learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07).* Hyderabad, India.

De Raedt, L. (2008). *Logical and relational learning*. Berlin: Springer Verlag.

De Raedt, L., & Dehaspe, L. (1997). Clausal discovery. *Machine Learning*, *26*, 99–146.

Dehaspe, L. (1997). Maximum entropy modeling with clausal constraints. In *Proceedings of the 7th International Workshop on Inductive Logic Programming (ILP-97).* Prague, Czech Republic.

Della Pietra, S., Della Pietra, V. J., & Lafferty, J. D. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(4), 380–393.

Dhillon, I., Mallela, S., & Modha, D. (2003). Information-theoretic co-clustering. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003).* Washington, DC.

Dou, Z., Song, R., & Wen, J. (2007). A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the Sixteenth International World Wide Web Conference (WWW-07).* Banff, Canada.

Duboc, A. L., Paes, A., & Zaverucha, G. (2008). Using the bottom clause and mode declarations on FOL theory revision from examples. In *Proceedings of the 18th International Conference on Inductive Logic Programming (ILP-08).* Prague, Czech Republic.

Džeroski, S., & Lavrač, N. (Eds.). (2001). *Relational data mining*. Berlin: Springer Verlag.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, *41*(1), 1–63.

Fitzpatrick, L., & Dent, M. (1997). Automatic feedback using past queries: social searching? In *Proceedings of 20th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-97).* Philadelphia, PA.

Forbus, K. D., & Oblinger, D. (1990). Making SME greedy and pragmatic. In *Proceedings of the Twelfth Annual Meeting of the Cognitive Science Society (CogSci-90).*

Fox, S., Karnawat, K., Mydland, M., Dumais, S. T., & White, T. (2005). Evaluating implicit measures to improve the search experience. *ACM Transactions on Information Systems*, *23*(2), 147–168.

Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. (2001). Learning probabilistic relational models. In S. Džeroski & N. Lavrač (Eds.), *Relational data mining.*

Getoor, L., & Taskar, B. (Eds.). (2007). *Introduction to statistical relational learning*. Cambridge, MA: MIT Press.

Goldsmith, J., & Sloan, R. H. (2005). New Horn revision algorithms. *Journal of Machine Learning Research*, *6*, 1919–1938.

Guestrin, C., Koller, D., Gearhart, C., & Kanodia, N. (2003). Generalizing plans to new environments in relational MDPs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03).* Acapulco, Mexico.

Hammersley, J., & Clifford, P. (1971). *Markov fields on graphs and lattices.* (Unpublished manuscript)

Heckerman, D. (1995). *A tutorial on learning Bayesian networks* (Tech. Rep. No. MSR-TR-95-06). Redmond, WA: Microsoft Research.

Herlocker, J., Konstan, J., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 230–237). Berkeley, CA.

Hinton, G. E. (2000). *Training products of experts by minimizing contrastive divergence* (Tech. Rep. No. GCNU TR 2000-004). Gatsby Computational Neuroscience Unit, University College London.

Huynh, T., & Mooney, R. (2008). Discriminative structure and parameter learning for Markov logic networks. In *Proceedings of the 25th International Confer-*

ence on Machine Learning (ICML-08) (pp. 416–423). Helsinki, Finland.

Huynh, T., & Mooney, R. J. (2009). Max-margin weight learning for Markov logic networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-09).* Bled, Slovenia. (To appear.)

Jansen, B. J., & Spink, A. (2006). How are we searching the World Wide Web? A comparison of nine search engine transaction logs. *Information Processing and Management*, *42*, 248–263.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002).* Edmonton, Canada.

Joachims, T., Granka, L., Pan, B., Hembrooke, H., & Gay, G. (2005). Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of 28th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-05).* Salvador, Brazil.

Kautz, H., Selman, B., & Jiang, Y. (1997). A general stochastic approach to solving problems with hard and soft constraints. In D. Gu, J. Du, & P. Pardalos (Eds.), (Vol. 35, pp. 573–586). American Mathematical Society.

Kersting, K., & De Raedt, L. (2001). Towards combining inductive logic programming with Bayesian networks. In *Proceedings of the 11th International Conference on Inductive Logic Programming (ILP-01)* (pp. 118–131). Strasbourg, France.

Kok, S., & Domingos, P. (2005). Learning the structure of Markov logic networks. In *Proceedings of 22nd International Conference on Machine Learning (ICML-05).* Bonn, Germany.

Kok, S., & Domingos, P. (2007). Statistical predicate invention. In *Proceedings of the 24th International Conference on Machine Learning (ICML-07).* Corvallis, OR.

Kok, S., & Domingos, P. (2009). Learning Markov logic network structure via hypergraph lifting. In *Proceedings of the 26th International Conference on Machine Learning (ICML-09).* Montreal, Quebec.

Kok, S., Singla, P., Richardson, M., & Domingos, P. (2005). *The Alchemy system for statistical relational AI* (Tech. Rep.). Department of Computer Science and Engineering, University of Washington. (`http://alchemy.cs.washington.edu/`)

Krause, A., & Horvitz, E. (2008). A utility-theoretic approach to privacy and personalization. In *Proceedings of the Twenty-Third Conference on Artificial*

Intelligence (AAAI-08). Chicago, IL.

Lavrač, N., & Džeroski, S. (1994). *Inductive logic programming: Techniques and applications*. Ellis Horwood.

Lee, S., Ganapathi, V., & Koller, D. (2007). Efficient structure learning of Markov networks using L_1-regularization. In *Advances in Neural Information Processing Systems 19 (NIPS-06)* (pp. 817–824).

Lesh, N., & Etzioni, O. (1995). A sound and fast goal recognizer. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95).* Montreal, Quebec.

Li, X., Wang, Y., & Acero, A. (2008). Learning query intent from regularized click graphs. In *Proceedings of 31st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-08).* Singapore.

Liang, P., & Jordan, M. I. (2008). An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *Proceedings of the 25th International Conference on Machine Learning (ICML-08).* Helsinki, Finland.

Lieberman, H. (1995). Letizia: An agent that assists web browsing. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95).* Montreal, Quebec.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematic Programming*, *45*(3), 503–528.

Liu, Y., & Stone, P. (2006). Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the Twenty-First Conference on Artificial Intelligence (AAAI-06).* Boston, MA.

Lowd, D., & Domingos, P. (2007). Efficient weight learning for Markov logic networks. In *Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-07).* Warsaw, Poland.

Manning, C. D., Raghavan, P., & Schutze, H. (2008). *Introduction to information retrieval*. Cambridge, England: Cambridge University Press.

Margaritis, D., & Thrun, S. (2000). Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems 13 (NIPS-00).* Denver, CO.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)* (pp. 187–192). Edmonton, Alberta.

Mihalkova, L., Huynh, T., & Mooney, R. J. (2007). Mapping and revising Markov logic networks for transfer learning. In *Proceedings of the Twenty-Second*

Conference on Artificial Intelligence (AAAI-07) (pp. 608–614). Vancouver, Canada.

Mihalkova, L., & Mooney, R. J. (2006). Transfer learning with Markov logic networks. In *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning.* Pittsburgh, PA.

Mihalkova, L., & Mooney, R. J. (2007). Bottom-up learning of Markov logic network structure. In *Proceedings of 24th International Conference on Machine Learning (ICML-07).* Corvallis, OR.

Mihalkova, L., & Mooney, R. J. (2008). Transfer learning by mapping with minimal target data. In *Proceedings of the AAAI workshop on Transfer Learning for Complex Tasks.* Chicago, IL.

Mihalkova, L., & Mooney, R. J. (2009a). Learning to disambiguate search queries from short sessions. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-09).* Bled, Slovenia. (To appear.)

Mihalkova, L., & Mooney, R. J. (2009b). Transfer learning from minimal target data by mapping across relational domains. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09).* Pasadena, CA.

Mihalkova, L., & Richardson, M. (2009). Speeding up inference in statistical relational learning by clustering similar query literals. In *Proceedings of the 19th International Conference on Inductive Logic Programming (ILP-09).* Leuven, Belgium.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing, 13*, 245–286.

Muggleton, S. (1996). Stochastic logic programs. In *Proceedings of the Sixth International Workshop on Inductive Logic Programming (ILP-96).* Stockholm, Sweden.

Muggleton, S., & Feng, C. (1992). Efficient induction of logic programs. In S. Muggleton (Ed.), *Inductive logic programming* (pp. 281–297). New York: Academic Press.

Niculescu-Mizil, A., & Caruana, R. (2005). Learning the structure of related tasks. In *Proceedings of NIPS-2005 Workshop on Inductive Transfer: 10 Years Later.* Whistler, Canada.

Niculescu-Mizil, A., & Caruana, R. (2007). Inductive transfer for Bayesian network structure learning. In *Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS-07).* San Juan, Puerto Rico.

Paes, A., Revoredo, K., Zaverucha, G., & Costa, V. S. (2005). Probabilistic first-order theory revision from examples. In *Proceedings of the 15th International Conference on Inductive Logic Programming (ILP-05).* Bonn, Germany.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Mateo, CA: Morgan Kaufmann.

Plotkin, G. D. (1970). A note on inductive generalisation. *Machine Intelligence*, *5*, 153–163.

Poon, H., & Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06).* Boston, MA.

Poon, H., & Domingos, P. (2007). Joint inference in information extraction. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07).* Vancouver, Canada.

Popescul, A., Ungar, L. H., Pennock, D. M., & Lawrence, S. (2001). Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of 17th Conference on Uncertainty in Artificial Intelligence (UAI-01).* Seattle, WA.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, *5*(3), 239–266.

Radlinski, F., & Joachims, T. (2005). Query chains: Learning to rank from implicit feedback. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-05).* Chicago, IL.

Raina, R., Ng, A. Y., & Koller, D. (2006). Constructing informative priors using transfer learning. In *Proceedings of 23rd International Conference on Machine Learning (ICML-06).* Pittsburg, PA.

Ramachandran, S., & Mooney, R. J. (1998). Theory refinement for Bayesian networks with hidden variables. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)* (pp. 454–462). Madison, WI.

Richards, B. L., & Mooney, R. J. (1992). Learning relations by pathfinding. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)* (pp. 50–55). San Jose, CA.

Richards, B. L., & Mooney, R. J. (1995). Automated refinement of first-order Horn-clause domain theories. *Machine Learning*, *19*(2), 95–131.

Richardson, M. (2004). *Learning and inference in collective knowledge bases*. Doctoral dissertation, University of Washington.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learn-*

ing, *62*, 107-136.

Russell, S., & Norvig, P. (2003). *Artificial intelligence: A modern approach* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

Sanderson, M. (2008). Ambiguous queries: Test collections need more sense. In *Proceedings of 31st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-08).*

Shavlik, J., & Natarajan, S. (2009). Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09).* Pasadena, CA.

Shen, D., Sun, J.-T., Yang, Q., & Chen, Z. (2006). Building bridges for Web query classification. In *Proceedings of 29th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-06).* Seattle, WA.

Shen, J., Li, L., Dietterich, T. G., & Herlocker, J. L. (2006). A hybrid learning system for recognizing user tasks from desktop activities and email messages. In *Proceedings of the 2006 International Conference on Intelligent User Interfaces (IUI-2006).* Miami, FL.

Silver, D., et al. (Eds.). (2005). *NIPS workshop on "Inductive Transfer : 10 Years Later".* Whistler, Canada.

Singla, P., & Domingos, P. (2005). Discriminative training of Markov logic networks. In *Proceedings of the Twentieth Conference on Artificial Intelligence (AAAI-05).* Pittsburgh, PA.

Singla, P., & Domingos, P. (2006). Memory-efficient inference in relational domains. In *Proceedings of the Twenty-First Conference on Artificial Intelligence (AAAI-06).* Boston, MA.

Singla, P., & Domingos, P. (2008). Lifted first-order belief propagation. In *Proceedings of the Twenty-Third Conference on Artificial Intelligence (AAAI-08).* Chicago, IL.

Spirtes, P., Glymour, C., & Scheines, R. (2001). *Causation, prediction, and search*. MIT Press.

Srinivasan, A. (2001). The Aleph manual [Computer software manual]. (`http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/`)

Sugiyama, K., Hatano, K., & Yoshikawa, M. (2004). Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the Thirteenth International World Wide Web Conference (WWW-04)* (pp. 675–684). New York, NY.

Sun, J., Wang, X., Shen, D., Zeng, H., & Chen, Z. (2006). Mining clicktrough data for collaborative web search. In *Proceedings of the Fifteenth International World Wide Web Conference (WWW-06)* (pp. 947–948). Edinburgh, Scotland. (Poster)

Sun, J., Zeng, H., Liu, H., Lu, Y., & Chen, Z. (2005). CubeSVD: A novel approach to personalized web search. In *Proceedings of the Fourteenth International World Wide Web Conference (WWW-05).* Chiba, Japan.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of 18th Conference on Uncertainty in Artificial Intelligence (UAI-02)* (pp. 485–492). Edmonton, Canada.

Taylor, M. E. (2008). *Autonomous inter-task transfer in reinforcement learning domains*. Doctoral dissertation. (Also appears as Artificial Intelligence Laboratory Technical Report UT-AI-TR-08-5)

Taylor, M. E., Fern, A., & Driessens, K. (Eds.). (2008). *AAAI workshop on "Transfer Learning for Complex Tasks".* Chicago, IL.

Taylor, M. E., Kuhlmann, G., & Stone, P. (2008). Autonomous transfer for reinforcement learning. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-08).* Estoril, Portugal.

Taylor, M. E., Stone, P., & Liu, Y. (2005). Value functions for RL-based behavior transfer: A comparative study. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05).* Pittsburgh, PA.

Taylor, M. E., Whiteson, S., & Stone, P. (2007). Transfer via inter-task mappings in policy search reinforcement learning. In *The sixth international joint conference on autonomous agents and multiagent systems (AAMAS-07).* Honolulu, HI.

Teevan, J., Dumais, S. T., & Horvitz, E. (2005). Personalizing search via automated analysis of interests and activities. In *Proceedings of 28th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-05).* Salvador, Brazil.

Teevan, J., Dumais, S. T., & Liebling, D. J. (2008). To personalize or not to personalize: Modeling queries with variation in user intent. In *Proceedings of 31st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-08).* Singapore.

Thomas, B. (2003). Bottom-up learning of logic programs for information extraction from hypertext documents. In *Proceedings of 7th European Conference of Principles and Practice of Knowledge Discovery in Databases (PKDD-03)* (pp. 435–446). Cavtat-Dubrovnik, Croatia.

Thrun, S., & O'Sullivan, J. (1996). Discovering structure in multiple learning tasks: The TC algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96).* Bari, Italy.

Thrun, S., & Pratt, L. (Eds.). (1998). *Learning to learn*. Boston: Kluwer Academic Publishers.

Torrey, L. (2009). *Relational transfer in reinforcement learning*. Doctoral dissertation, University of Wisconsin - Madison.

Torrey, L., & Shavlik, J. (2009). Policy transfer via Markov logic networks. In *Proceedings of the 19th International Conference on Inductive Logic Programming (ILP-09).* Leuven, Belgium.

Torrey, L., Shavlik, J., Natarajan, S., Kuppili, P., & Walker, T. (2008). Transfer in reinforcement learning via Markov logic networks. In *Proceedings of the AAAI workshop on Transfer Learning for Complex Tasks.* Chicago, IL.

Torrey, L., Shavlik, J., Walker, T., & Maclin, R. (2007). Relational macros for transfer in reinforcement learning. In *Proceedings of the 17th International Conference on Inductive Logic Programming (ILP-07).* Corvallis, OR.

Torrey, L., Walker, T., Shavlik, J., & Maclin, R. (2005). Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Proceedings of the 16th European Conference on Machine Learning (ECML-05).* Porto, Portugal.

Wang, X., & Zhai, C. (2007). Learn from web search logs to organize search results. In *Proceedings of 30th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-07).* Amsterdam, The Netherlands.

Yue, Y., & Joachims, T. (2008). Predicting diverse subsets using structural SVMs. In *Proceedings of the 25th International Conference on Machine Learning (ICML-08)* (pp. 1224–1231). Helsinki, Finland.

Zelle, J. M., Mooney, R. J., & Konvisser, J. B. (1994). Combining top-down and bottom-up methods in inductive logic programming. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML-94)* (pp. 343–351). New Brunswick, NJ.

Vita

Lilyana Mihalkova, who goes by Lily, was born in 1981 in Yambol, Bulgaria, but went to school in Pazardjik, Bulgaria, where her family still lives. In 1994, after she sat for exams in mathematics and Bulgarian literature, she was accepted to the “Bertolt Brecht” Language School where, in addition to the regular subjects, she studied English and German. In the fall of 1999, Lily started her college education at the American University in Bulgaria in Blagoevgrad, Bulgaria, where she spent two years and met some wonderful friends. She continued her undergraduate studies at Hope College in Holland, MI, from where she received her Bachelor of Science degree in 2003. In the fall of 2003, she started her doctoral studies at the University of Texas at Austin. Halfway through her dissertation work, Lily spent a fun summer at Microsoft Research in Redmond, WA. After graduation, Lily will join the University of Maryland, College Park as a post-doc.

Permanent address: 34 Gavril Krustevich Str. apt. 11
Pazardjik 4400
Bulgaria

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth’s TeX Program.