KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Relational Approaches for Learning, Transferring and Mining

Jan Van Haaren

Supervisor:
Prof. dr. J. Davis

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

December 2016

Relational Approaches for Learning, Transferring and Mining

Jan VAN HAAREN

Examination committee:
Prof. dr. A. Bultheel, chair
Prof. dr. J. Davis, supervisor
Prof. dr. ir. H. Blockeel
Prof. dr. ir. E. Steegmans
Prof. dr. Pascal Fua
(École Polytechnique Fédérale de Lausanne, Switzerland)
Prof. dr. Ulf Brefeld
(Leuphana Universität Lüneburg, Germany)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

December 2016

Uitgegeven in eigen beheer, Jan Van Haaren, Celestijnenlaan 200A box 2402, B-3001 Leuven (Belgium)

Acknowledgements

For a nostalgic like me, writing this acknowledgements section is a true joy. Not only because it marks the end of my PhD but foremost because it provides an excellent occasion to look back at the past five years of my life. My PhD has been a fun, challenging, rewarding, and especially self-enriching experience in many respects. It provided me with the opportunity to work with many wonderful and inspiring people whom I would like to thank explicitly.

I am very grateful to my supervisor professor Jesse Davis for countless reasons. As one of his first recruits, I have had the privilege of working with him very closely in the early days of my PhD. This close collaboration has not only helped me in becoming a better researcher but also in getting to know and understand the academic scene. As Jesse's group started to grow throughout the years, he gave me more responsibilities and freedom to explore and develop my own ideas, while still being actively involved in most of my scientific undertakings. I cannot thank him enough for all his precious feedback and advice, which I will certainly carry with me in my future endeavors. I look forward to continue to work with him in the future on interesting and challenging sports problems.

I would like to thank professors Hendrik Blockeel, Eric Steegmans, Pascal Fua, and Ulf Brefeld for serving on my examination committee and professor

Adhemar Bultheel for chairing my preliminary and public PhD defenses. I am thankful for their insightful questions and constructive feedback on earlier versions of this dissertation. I wish to thank Pascal in particular for offering me the opportunity to do an internship in his lab. My stay in Lausanne was a great experience that even lead to one of the contributions in this dissertation. Furthermore, I also wish to thank the Agency for Innovation by Science and Technology in Flanders (IWT), the Research Fund KU Leuven (CREA/11/015), and the European Commission (FP7 Marie Curie Career Integration Grant #294068) for their financial support.

Although my name appears first on all the publications included in this dissertation, this work would not have been possible without the help of my co-authors. In particular, I am very grateful to Guy Van den Broeck, Wannes Meert, Andrey Kolobov, and Vladimir Dzyuba. They are not only excellent researchers whom I have learned a lot from but also awesome people to work with. Furthermore, I also enjoyed working with Tim Op De Beéck, Toon Van Craenendonck, Anton Dries, Vincent Vercruyssen, Jan Lasek, Tom Decroos, Albrecht Zimmermann, and Mehdi Kaytoue on other exciting sports projects.

I would like to express my gratitude to all members of the Machine Learning group in Leuven and the Computer Vision lab in Lausanne for providing a pleasant and inspiring work environment. I have had the pleasure of sharing an office with Irma Ravkić, Tim Op De Beéck, Daan Fierens, Dimitar Shterionov, Tom Decroos, and Amaury Dame. I truly appreciate their feedback, help, and support. In particular, I wish to explicitly thank Daan for being an excellent mentor and a source of inspiration at the start of my PhD.

Furthermore, I enjoyed the pleasant lunch-time conversations, which were often a welcome distraction from work, with my former and current colleagues in Leuven and Lausanne, including Aäron Verachtert, Agata Mosinska, Amaury Dame, Angelika Kimmig, Anna Latour, Antoine Adam, Anton Dries, Behrouz Babaki, Benjamin Negrevergne, Bogdan Moldovan, Carlos Becker, Celine Vens, Clément Charnay, Daan Fierens, Daniele Alfarone, Davide Nitti, Denny Verbeeck, Eduard Trulls, Eduardo Costa, Elia Van Wolputte, Evgeniya Korneva, Francesco Orsini, Gitte Vanwinckelen, Guy Van den Broeck, Hendrik Blockeel, Horesh Ben Shitrit, Irma Ravkić, Jessa Bekker, Joana Côrte-Real, Jonas Vlasselaer, Joris Renkens, Kaja Zupanc, Kilian Hendrickx, Ksenia Konyushkova, Kurt De Grave, Kwang Moo Yi, Leander Schietgat, Luc De Raedt, Martin Žnidaršič, Mathias Verbeke, Mathieu Salzmann, Matthijs van Leeuwen, McElory Hoffmann, Pablo Márquez, Pedro Zuidberg, Róger Bermúdez, Samuel

Kolb, Sebastijan Dumančić, Sergey Paramonov, Siegfried Nijssen, Steffen Michels, Thanh Le Van, Tias Guns, Tim Op De Beéck, Tom Decroos, Toon Van Craenendonck, Vaishak Belle, Vincent Nys, Vincent Vercruyssen, Vladimir Dzyuba, Xinchao Wang, and Yannick Verdie.

My special thanks goes to my friends and family. In addition to my team mates and coaches at my football club *KSK Weelde* and my quiz team *De LAT-relatie*, I would like to thank Alexander van den Berghe, Harm Leenders, Michaël Derde, Nicky Verheijen, Robin Peeters, Thomas Lefebvre, Thomas Plas, and Valérie Van Damme for their support and the many great moments together.

Last but not least, the people I am particularly grateful to are my loving and caring parents Ria and Jef, whom I cannot thank enough for their emotional and financial support. I will be eternally grateful for the way they raised me, and for allowing me to go my own way and to chase my dreams. I also wish to thank my sister Sanne and her boyfriend Toon as well as my grandparents, uncles, and aunts for showing sincere interest in my work and achievements.

Jan Van Haaren
Heverlee, December 2016

Abstract

Machine learning aims to design algorithms whose performance on a task improves with experience, where experience is usually defined as the amount of available data. Unfortunately, most traditional machine-learning algorithms rely on assumptions that do not hold in important real-world applications. One assumption is that the data have a simple structure, while most real-world problems involve complex, relational data. Another assumption is that large quantities of data are available to learn a sufficiently accurate predictive model, while high-quality data are often scarce in real-world domains.

This dissertation aims to overcome the limitations of traditional machine-learning algorithms by proposing approaches for learning predictive models in domains that are characterized by a complex, relational structure and a shortage of high-quality data to learn accurate models. Furthermore, this dissertation also applies relational-learning techniques to spatio-temporal sports data, which are illustrative for the challenges that many other real-world applications pose.

Most of the work in this dissertation relates to the field of statistical relational learning, which is concerned with learning predictive models for domains exhibiting both uncertainty and a complex structure. To address the inherent challenges, statistical-relational-learning formalisms typically combine a prob-

abilistic model with a relational representation. This dissertation leverages Markov logic networks, which combine Markov random fields with logic.

This dissertation presents five main contributions. The first contribution is an algorithm for learning Markov random fields from binary data. The second contribution is an algorithm for learning Markov logic networks from relational data. The third contribution is an algorithm for transferring knowledge across relational domains, where the domains can be entirely different. The fourth contribution is an approach for discovering offensive strategies in spatio-temporal soccer match data. The fifth contribution is an approach for discovering offensive patterns in spatio-temporal volleyball match data.

Beknopte samenvatting

Automatisch leren streeft ernaar om algoritmes te ontwerpen waarvan de prestaties op een taak verbeteren naargelang zij ervaring opdoen, waarbij ervaring meestal gedefinieerd wordt als de hoeveelheid beschikbare gegevens. De meeste traditionele automatische leeralgoritmes steunen echter op aannames die helaas niet gelden voor belangrijke toepassingen in de echte wereld. Eén aanname is dat de gegevens een eenvoudige structuur hebben, terwijl de meeste problemen in de echte wereld complexe, relationele gegevens met zich meebrengen. Een andere aanname is dat grote hoeveelheden gegevens beschikbaar zijn om een voldoende nauwkeurig voorspellend model te leren, terwijl kwalitatieve gegevens vaak schaars zijn in echte domeinen.

Dit proefschrift streeft ernaar om de beperkingen van traditionele automatische leeralgoritmes te overwinnen door benaderingen voor te stellen om voorspellende modellen te leren in domeinen die gekenmerkt worden door een complexe, relationele structuur en een tekort aan kwalitatieve gegevens om nauwkeurige modellen te leren. Dit proefschrift past daarnaast ook technieken voor relationeel leren toe op spatio-temporele sportgegevens die illustratief zijn voor de uitdagingen die vele andere echte toepassingen bieden.

Het meeste werk in dit proefschrift heeft betrekking tot het veld van statistisch relationeel leren dat zich toelegt op het leren van voorspellende modellen in domeinen die gekenmerkt worden door zowel onzekerheid als een complexe structuur. Benaderingen voor statistisch relationeel leren combineren meestal een probabilistisch model met een relationele voorstelling om de inherente uitdagingen aan te pakken. Dit proefschrift maakt gebruik van *Markov logic networks* die *Markov random fields* met logica combineren.

Dit proefschrift stelt vijf hoofdbijdrages voor. De eerste bijdrage is een algoritme om *Markov random fields* te leren uit binaire gegevens. De tweede bijdrage is een algoritme om *Markov logic networks* te leren uit relationele gegevens. De derde bijdrage is een algoritme om kennis over te dragen tussen relationele domeinen, waarbij de domeinen volledig verschillend kunnen zijn. De vierde bijdrage is een benadering om aanvallende strategieën in spatio-temporele voetbalgegevens te ontdekken. De vijfde bijdrage is een benadering om aanvallende patronen in spatio-temporele volleybalgegevens te ontdekken.

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

The field of machine learning aims to design algorithms that enable computers to learn without being explicitly programmed. Like humans, computers are said to learn if their performance on a task improves as they gain more experience. The current paradigm in machine learning defines experience as the amount of available training data. Hence, machine-learning algorithms ideally produce more accurate predictive models by processing more data. However, traditional algorithms typically rely on a number of assumptions that often do not hold in practice and restrict their applicability to important real-world problems.

One limitation of standard machine-learning algorithms is that they operate on simply-structured data. Most traditional algorithms expect the data to be tabular, where each row represents an entity and each column corresponds to an attribute. However, many *real-world domains involve complexly-structured data*. These data comprise different types of entities, which can have different attributes, and different relationships between these entities. For example, in a soccer domain, we wish to store distinct attributes for players, managers, and clubs. In addition, we also want to express different types of relationships between these entities. For example, we wish to model that a player plays for a certain club, or that a manager manages a certain player.

The field of *statistical relational learning* (SRL, De Raedt et al. 2008; Getoor and Taskar 2007) aims to upgrade machine-learning approaches to handle complex data. Statistical relational learning focuses on learning compact models that allow reasoning in domains involving uncertainty and complex data. To this end, statistical-relational-learning formalisms try to exploit the relational structure of the data. One challenge is to automatically learn the models from data. Another challenge is to ensure that the learned models allow efficient reasoning.

Another limitation of standard machine-learning algorithms is that they assume large quantities of training data are available, while *high-quality data are often scarce or unavailable in real-world domains* to learn a sufficiently accurate predictive model. High-quality training data can often be time consuming and expensive to collect or even impossible to obtain at all. Moreover, whenever training data are available in large quantities, the data are often inconsistent and noisy. For example, in a soccer domain, the data collection process during matches requires teams of human annotators in addition to expensive optical-tracking systems (Bialik 2014b). While tedious and time consuming, the manual annotation of data can easily lead to inconsistencies and errors (Bialik 2014a).

The field of *transfer learning* (TL, Pan and Yang 2010) aims to overcome the dependence of traditional machine-learning approaches on large quantities of high-quality data. Transfer learning aims to learn accurate predictive models in domains where data are scarce by leveraging data from other related domains. One challenge is to automatically discover knowledge that can be transferred from one domain to the other. Another challenge is to represent the discovered knowledge in a domain-independent way as different domains can consist of different entities and relationships.

The high-level objective of this dissertation is developing novel algorithms and approaches that enable machine-learning techniques to be applied to important real-world problems. More specifically, the work in this dissertation concerns learning predictive models in domains involving uncertainty, learning predictive models in domains where training data are scarce, and learning predictive models that allow for efficient reasoning. In addition, this dissertation presents applications of relational approaches to spatio-temporal sports data, which are illustrative for the challenges that many real-world domains pose.

Despite the restricted applicability of traditional machine-learning algorithms to real-world problems, several machine-learning successes have appeared in the newspapers recently. Examples of notable success stories include autonomous

vehicles (e.g., Kang 2016), personal voice assistants in mobile devices (e.g., Chen 2016), and AlphaGo (Silver et al. 2016), which was the first computer program to defeat a human professional player in the ancient game of Go. Most of the recent successes in machine learning are powered by deep-learning techniques (LeCun et al. 2015). However, the necessary conditions for deep-learning algorithms to be successful are limiting in real-world applications.

Deep learning's appetite for huge quantities of training data and inability to learn interpretable models prevent the paradigm from being applicable to a wider range of important real-world problems. Therefore, the machine-learning community should keep exploring innovations that alleviate these limitations as well as alternative paradigms. This dissertation adds to that by presenting novel algorithms that can handle complex relational data in a natural way as well as learn from small quantities of high-quality training data. Although developed under a different paradigm and valuable on their own, the insights from this dissertation can likely also inspire future deep-learning research.

1.1 Dissertation Statement

This dissertation presents novel relational approaches that allow machine learning to be applied to important real-world problems, which are typically characterized by uncertainty, complexly-structured data, and a limited amount of high-quality training data. This dissertation also explores the application of relational approaches to spatio-temporal sports data.

1.2 Contributions

This dissertation presents five main contributions. The first three contributions are relevant to the domain of statistical relational learning, while the remaining two contributions are relevant to the field of sports analytics.

1.2.1 Statistical Relational Learning

The field of statistical relational learning (SRL, De Raedt et al. 2008; Getoor and Taskar 2007) is concerned with domains involving uncertainty on one hand

and complexly-structured data on the other hand. Statistical-relational-learning formalisms typically leverage a probabilistic model to handle the uncertainty and use a relational representation to model the complexly-structured data. This dissertation relies on Markov logic networks, which use Markov random fields as the probabilistic model and first-order logic as the relational representation.

Structure Learning of Markov Random Fields

The first main contribution of this dissertation is a novel algorithm called GSSL (**G**enerate **S**elect **S**tructure **L**earning) for automatically learning Markov random fields from binary data. The algorithm, which combines some of the benefits of existing structure-learning approaches, views structure learning as a feature-induction problem. GSSL first quickly generates a large set of candidate features in a data-driven way and then selects a subset of the generated features to include in the final model.

An extensive empirical evaluation on 20 real-world datasets demonstrates the advantages of the proposed algorithm. In addition to learning more accurate models than the baseline approaches, GSSL is also much faster.

The Java source code of the GSSL implementation is available for download from `https://dtai.cs.kuleuven.be/software/gssl`.

Lifted Structure Learning of Markov Logic Networks

The second main contribution of this dissertation is a novel algorithm called LSL (**L**ifted **S**tructure **L**earning) for automatically learning tractable Markov logic networks from relational data. The algorithm, which performs a greedy search in the space of candidate models, leverages techniques from lifted inference to optimize the exact training-set log-likelihood and to learn models that are guaranteed to support certain types of queries efficiently.

An extensive empirical evaluation on three real-world datasets demonstrates the advantages of the proposed algorithm. LSL learns more accurate models than the baseline approaches in terms of both generative and predictive quality. Moreover, LSL even outperforms off-the-shelf structure-learning approaches that learn intractable models on prediction tasks, suggesting that tractable models are a powerful hypothesis space that is sufficient for many problems.

The Java source code of the LSL implementation is available for download from https://dtai.cs.kuleuven.be/software/wfomc.

Deep Transfer Learning in Relational Domains

The third main contribution of this dissertation is a novel framework called TODTLER (**T**wo-**O**rder-**D**eep **T**ransfer **Lea**r**n**ing) for performing deep-transfer learning in relational domains. The framework views knowledge transfer as the process of first learning a declarative bias in one domain and then applying that bias to another domain to improve the learning process.

The concrete implementation of the framework performs deep-transfer learning, where the source and target domains can consist of entirely different sets of entities and relationships, in the context of Markov logic networks. TODTLER represents the transferable knowledge as a distribution over domain-independent second-order templates, which give rise to first-order Markov logic formulas in a particular domain.

An extensive empirical evaluation on three real-world datasets shows that the algorithm outperforms the state-of-the-art deep-transfer-learning algorithm as well as the state-of-the-art inductive-learning algorithm in terms of accuracy. In addition to learning more accurate models, TODTLER is also much faster than the existing deep-transfer-learning algorithms.

The Java source code of the TODTLER implementation is available for download from https://dtai.cs.kuleuven.be/software/todtler.

1.2.2 Sports Analytics

The broad field of sports analytics (SA, Alamar 2013) is concerned with the use of data in the sports industry. Since the field lacks a formal definition, almost any research in the context of sports that leverages data is considered sports analytics nowadays. However, this dissertation adopts a more restrictive definition, which views sports analytics as leveraging data to improve the performance of individual athletes and sports teams.

Discovering Offensive Strategies in Soccer Match Data

The fourth main contribution of this dissertation is an inductive-logic-programming approach that automatically discovers offensive strategies in spatio-temporal soccer match data. The approach aims to discover patterns that frequently occur in situations leading to a goal attempt.

An empirical study on a large number of matches from a Belgian professional soccer club demonstrates that the proposed approach is able to automatically discover interesting and relevant offensive strategies in match data.

Discovering Offensive Patterns in Volleyball Match Data

The fifth main contribution of this dissertation is an inductive-logic-programming approach that automatically discovers offensive patterns in spatio-temporal volleyball match data. The approach aims to discover patterns that occur frequently in won rallies and infrequently in lost rallies. In addition, the approach also aims to reveal patterns that are used by one team in a particular match but not the opposing team.

An analysis of both the men's and women's final match at the 2014 FIVB Volleyball World Championships demonstrates that the proposed approach is able to discover meaningful offensive patterns in volleyball match data.

1.3 Structure of the Dissertation

The structure of this dissertation is as follows.

Chapter 2 provides the necessary background on logic and inductive logic programming, Markov random fields, Markov logic networks, and transfer learning. This chapter is based on the publications cited in chapters 3 to 7.

Chapter 3 introduces a novel algorithm for automatically learning Markov random fields from data and presents an extensive empirical evaluation on 20 real-world datasets. This chapter is based on the following publication:

Jan Van Haaren and Jesse Davis (2012). "Markov Network Structure Learning: A Randomized Feature Generation Approach". In: *Proceedings of the Twenty-Sixth*

AAAI Conference on Artificial Intelligence (AAAI 2012; Toronto, Ontario, Canada; 22-26 July 2012), pages 1148–1154

Chapter 4 introduces a novel algorithm for automatically learning tractable Markov logic networks from data and presents an extensive empirical evaluation on three real-world datasets. This chapter is based on the following publications:

Jan Van Haaren, Guy Van den Broeck, Wannes Meert, and Jesse Davis (2016). "Lifted Generative Learning of Markov Logic Networks". In: *Machine Learning* 103(1), pages 27–55

Jan Van Haaren, Guy Van den Broeck, Wannes Meert, and Jesse Davis (2014c). "Tractable Learning of Liftable Markov Logic Networks". In: *Proceedings of the ICML 2014 Workshop on Learning Tractable Probabilistic Models (LTPM 2014; Beijing, China; 26 June 2014)*

Chapter 5 introduces a novel framework for performing deep-transfer learning in relational domains and presents an extensive empirical evaluation on three real-world datasets. This chapter is based on the following publication:

Jan Van Haaren, Andrey Kolobov, and Jesse Davis (2015). "TODTLER: Two-Order-Deep Transfer Learning". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015; Austin, Texas, United States; 25-30 January 2015)*, pages 3007–3015

Chapter 6 introduces an approach for automatically discovering offensive strategies in spatio-temporal soccer match data and presents an empirical study on a large volume of matches from a Belgian professional soccer club. This chapter is based on the following publication:

Jan Van Haaren, Vladimir Dzyuba, Siebe Hannosset, and Jesse Davis (2015). "Automatically Discovering Offensive Patterns in Soccer Match Data". In: *Advances in Intelligent Data Analysis XIV (IDA 2015; Saint-Étienne, France; 22-24 October 2015)*, pages 286–297

Chapter 7 introduces an approach for automatically discovering offensive patterns in spatio-temporal volleyball match data and presents an analysis of both the men's and women's final match at the 2014 FIVB Volleyball World Championships. This chapter is based on the following publication:

Jan Van Haaren, Horesh Ben Shitrit, Jesse Davis, and Pascal Fua (2016). "Analyzing Volleyball Match Data from the 2014 World Championships Using Machine Learning Techniques". In: *Proceedings of the Twenty-Second ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016; San Francisco, California, United States; 13-17 August 2016)

Chapter 8 summarizes the contributions presented in this dissertation and provides promising avenues for future research.

2

Background

This chapter provides the necessary background on logic and inductive logic programming, Markov random fields, Markov logic networks, and transfer learning that this dissertation relies on.

2.1 Logic and Inductive Logic Programming

Logic is a commonly used representation language for relational data. Inductive logic programming (ILP, Džeroski and Lavrač 2001) is a well-known framework for learning models from relational data.

2.1.1 Logic

Logic is a powerful and expressive language. This section introduces the relevant subsets of first-order and second-order logic that we use in this dissertation.

First-Order Logic

We use a subset of *first-order logic* with an alphabet consisting of three symbols. A *constant* starts with a lowercase letter and refers to specific objects in a domain. For example, the constant alice refers to a person called Alice. A *variable* starts with an uppercase letter and ranges over multiple objects in a domain. For example, the variable People refers to a group of people. A *predicate* either represents a relation between objects in a domain or defines a property of an object. For example, the predicate Friends indicates that two people are friends, while the predicate Mother indicates that a person is someone's mother.

Using these three symbols, we define the following constructs. A *term* is either a variable or a constant. An *atom* is a predicate applied to one or multiple terms. For example, the atom $\mathsf{Friends}(\mathsf{alice}, \mathsf{bob})$ expresses that Alice and Bob are friends. A *literal* is either an atom or its negation. A *formula* is a finite set of literals that are connected using the conjunctive operator $\wedge$ or the disjunctive operator $\vee$. For example, the formula $\mathsf{Friends}(\mathsf{alice}, \mathsf{bob}) \wedge \mathsf{Friends}(\mathsf{bob}, \mathsf{carol})$ expresses that Alice is friends with Bob and that Bob is friends with Carol. A *clause* is a formula, where the literals are solely connected using the disjunctive operator $\vee$. For example, the formula $\mathsf{Friends}(\mathsf{alice}, \mathsf{bob}) \vee \mathsf{Siblings}(\mathsf{alice}, \mathsf{bob})$ expresses that Alice and Bob are friends or siblings. A *definite clause* is a clause that has exactly one positive literal. A *definite program* is a collection of definite clauses. A term, atom, or formula are *ground* if they only contain constants.

Second-Order Logic

We use a limited form of *second-order logic* that augments the alphabet of our subset of first-order logic with one additional symbol. A *predicate variable* is a variable that ranges over multiple predicates in a domain. For example, in the second-order literal $\mathsf{P}(\mathsf{alice}, \mathsf{bob})$ the predicate variable P can take the name of any predicate in the domain (e.g., Friends or Siblings).

2.1.2 Inductive Logic Programming

This section introduces the inductive-logic-programming learning task and presents the widely-adopted Aleph system.

Definition

Inductive logic programming allows to directly model important relationships and facilitates incorporating domain knowledge into the learning process. Informally, ILP attempts to learn a definite program that, in combination with background knowledge, can be used to distinguish positive and negative examples. More formally, we define the ILP learning task as follows:

Given: A target predicate T, background knowledge BK, a non-empty set of *positive* examples $E+$ of T, and a set of *negative* examples $E-$ of T.

Learn: A set of definite clauses S such that $BK \wedge S \models E^+$ and $BK \wedge S \not\models E^-$.

In practice, it is often not possible to ensure $BK \wedge S \not\models E^-$. Hence, this condition is typically relaxed by permitting clauses in S to cover negative examples as well. A clause covers an example if the clause, in combination with BK, can be used to derive that the target predicate T is true for the given example. The goal in the relaxed setting is to achieve a balance between the number of positive and negative examples that each clause covers.

Aleph

In this dissertation, we employ the widely-used Aleph ILP system (Srinivasan 2001) to address the above task. Aleph learns a set of definite clauses by iteratively applying the following two-step approach.

In the *saturation* step, the system first selects a random positive example, called the seed example, and finds all facts in the background knowledge that are true for this example. Aleph forms a clause where the body is the conjunction of all these true facts and the head is the target predicate. This is the most-specific clause, or bottom clause, that covers the seed example.

In the *search* step, the system performs a top-down search over clause bodies that generalize the bottom clause. The key idea is that a subset of the facts can be used to explain the seed example's label and that this explanation is likely to apply to other examples as well.

2.2 Markov Random Fields

A Markov random field is a compact representation of a joint probability distribution over multiple variables. This section introduces the relevant aspects of representation, inference, and learning for Markov random fields.

2.2.1 Representation

A *Markov random field* (MRF) or *Markov network* (MN) is a graphical model that compactly represents a joint probability distribution over a collection of variables $X = (X_1, X_2, \ldots, X_n)$, where n is the number of variables (Della Pietra et al. 1997). A Markov random field consists of an undirected graph G and a set of potential functions ϕ_k. The graph contains a node for each variable and the model has a potential function for each clique in the graph. A Markov random field represents the following joint probability distribution:

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

where $x_{\{k\}}$ is the state of the variables that appear in the k-th clique, and Z is a normalization constant.

Markov random fields are often conveniently represented as *log-linear models*, where each clique potential is replaced by an exponentiated weighted sum of features of the state of the variables:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(x)\right). \tag{2.2}$$

A feature $f_i(x)$ may be any real-valued function of the state of the variables. For discrete data, a feature typically is a conjunction of tests of the form $X_i = x_i$, where X_i is a variable and x_i is a value of that variable. A feature matches an example if it is true in that example.

2.2.2 Inference

The main inference task in graphical models is to compute the conditional probability of a set of variables $V = (V_1, V_2, \ldots, X_n)$ given the values assigned to another set of variables $W = (W_1, W_2, \ldots, W_n)$ by summing over all possible assignments to the remaining variables $U_i \notin V \cup W$. The set of variables V is called the *query*, while the set W is called the *evidence*.

Since computing conditional probabilities is a #P-complete problem, this task requires approximate inference techniques. One widely used method is Markov chain Monte Carlo (MCMC, Gilks et al. 1996) and Gibbs sampling in particular. Markov chain Monte Carlo proceeds by sampling each variable in turn given its *Markov blanket*, which is the set of variables that it appears with in a potential. Another popular method for approximate inference is Loopy Belief Propagation (LBP, Murphy et al. 1999).

2.2.3 Parameter Learning

The parameter learning task is to automatically learn the weights w_i associated with the features f_i from data by optimizing a given objective function. Ideally, each candidate Markov random field is scored by its training set log-likelihood (LL), which is a convex function of its weights. Hence, learning the optimal weights can be solved via convex optimization. However, this typically requires an iterative optimization technique where each step of the optimization must calculate both the log-likelihood and its gradient. This is often computationally infeasible as it requires computing the partition function Z in Equation 2.2. Moreover, Kulesza and Pereira (2008) have found that employing approximate inference can mislead parameter learning algorithms.

A more efficient alternative that has been widely used in domains such as spatial statistics, social network modeling and language processing is optimizing the pseudo-log-likelihood (PLL, Besag 1975). This approximation is defined as:

$$\log P_w^{\bullet}(X = x) = \sum_{i=1}^{V} \sum_{j=1}^{N} \log P_w(X_{j,i} = x_{j,i} | MB_x(X_{j,i})) \tag{2.3}$$

where V is the number of variables, N is the number of training examples, $x_{j,i}$

is the value of the i-th variable of the j-th example, $MB_x(X_{j,i})$ is the state of the Markov blanket of $X_{j,i}$ in the data. The pseudo-log-likelihood is much more efficient to compute than the actual log-likelihood and can also be optimized via convex optimization.

2.2.4 Structure Learning

The structure learning task is to automatically learn the features f_j as well as their corresponding weights w_j from data. The traditional approach to this task is to view structure learning as a feature induction problem. The existing approaches that address this task can be largely divided into two categories: search-based approaches and local-model-based approaches.

Search-based Approaches

The standard algorithm to Markov-random-field structure learning by Della Pietra et al. (1997) uses a greedy, general-to-specific (i.e., top-down) search. The algorithm starts with a set of atomic features, which correspond to the variables in the domain. It creates candidate features by conjoining each feature to each other feature, including the original atomic features. The algorithm evaluates each candidate feature f by estimating how much including f in the model would improve the pseudo-log-likelihood of the model. It adds the feature that results in the largest pseudo-log-likelihood gain to the feature set. The algorithm terminates when no candidate feature improves the pseudo-log-likelihood.

The more recent BLM algorithm by Davis and Domingos (2010) employs a greedy, specific-to-general (i.e., bottom-up) search. BLM starts by treating each complete example as a long feature in the Markov random field. The algorithm repeatedly iterates through the feature set and considers generalizing each feature to match its k nearest previously unmatched examples by dropping variables. If incorporating the newly generalized feature improves the pseudo-log-likelihood of the model, it is retained in the model. The process terminates when no generalization improves the pseudo-log-likelihood of the model.

The drawback of search-based approaches is that they must perform parameter learning to score each candidate feature. This is computationally expensive even when optimizing the pseudo-log-likelihood instead of the actual log-likelihood.

Local-Model-based Approaches

The more recent approaches to Markov-random-field structure learning first learn a set of local models and then combine them into a global model. At a high level, these algorithms proceed in two steps. In the first step, they aim to discover the Markov blanket of each variable X_i by building a model that predicts the value of X_i given the remaining variables. In the second step, they add all discovered features to the model and globally learn their corresponding weights using any standard weight learning algorithm.

The L1 algorithm by Ravikumar et al. (2010) employs L_1-logistic-regression models as local models. In the limit of infinite data, consistency is guaranteed such that X_i is in the Markov blanket of X_j if and only if X_j is in the Markov blanket of X_i. In practice, however, this is often not the case and there are two methods to decide which features to include in the model. The first method includes a feature if either X_i is in the Markov blanket of X_j or X_j is in the Markov blanket of X_i. The second method includes a feature if both X_i is in the Markov blanket of X_j and X_j is in the Markov blanket of X_i. A weakness of the L1 algorithm is that it only constructs pairwise features.

The DTSL algorithm by Lowd and Davis (2010) employs the same general strategy using a probabilistic decision-tree learner as local model. The algorithm first learns a probabilistic decision tree for each variable and then converts each tree into a set of conjunctive features. The most straightforward conversion method constructs one feature for each root-to-leaf path through the tree although the paper proposes several other conversion methods as well.

The drawback of local-model-based approaches is that it can be computationally expensive to learn the local models if the dataset contains a large number of variables or examples.

2.3 Markov Logic Networks

A Markov logic network combines Markov random fields with first-order logic to allow for probabilistic inference. This section introduces the relevant aspects of representation, inference, and learning for Markov logic networks.

2.3.1 Representation

A *Markov logic network* (MLN, Richardson and Domingos 2006) softens first-order logic by associating a weight to each formula. More formally, a Markov logic network is a collection of pairs (F_i, w_i), where each F_i is a first-order formula and $w_i \in \mathbb{R}$ its associated weight. As w_i increases, the strength of the constraint that F_i imposes on the world increases too. Thus, worlds that violate formulas become less likely but not impossible as is the case in first-order logic. Formulas with infinite weights represent pure first-order logic formulas.

A Markov logic network provides a template for constructing a Markov random field. For a given set of constants, which we call the *domain*, the MLN formulas define a Markov random field. The graph contains a node for each variable, which are the ground instances of the atoms that appear in the formulas. The edges in the graph connect the literals that appear in the same ground instance of a formula. A Markov logic network induces the following joint probability distribution over a relational database db:

$$P(db) = \frac{1}{Z} \exp\left(\sum_{i=1} w_i n_i(db)\right) \tag{2.4}$$

where w_i is the weight of the i-th formula, $n_i(db)$ is the number of true ground instances of formula F_i in database db, and Z is a normalization constant.

2.3.2 Inference

An important inference task in MLNs is to compute the conditional probability of a formula F_1 given the truth value of another formula F_2 by summing over the truth values of the remaining formulas F_i. The formula F_1 is called the *query*, while the formula F_2 is called the *evidence*.

Since a Markov logic network provides a template for constructing a Markov random field, the inference tasks in MLNs and MRFs are tightly connected with each other. The standard way of performing inference in an MLN is by first constructing the corresponding MRF for a given set of constants and then performing inference in the resulting MRF. For large domains, this approach

often leads to performing inference in very large and densely-connected MRFs causing inference to become intractable.

The popularity of statistical-relational-learning languages such as Markov logic networks has motivated a new class of *lifted inference* algorithms, which aim to speed up inference by exploiting the high-level structure and symmetries of the first-order logic formulas (Kersting 2012; Poole 2003). Van den Broeck (2011) introduces the notion of *domain-lifted inference* which formalizes the intuition that lifted inference algorithms should efficiently deal with large domains by running in time polynomial in the domain size. However, although domain-lifted inference algorithms should be polynomial in the number of constants, they can still be exponential in other parameters such as the number of formulas.

2.3.3 Parameter Learning

The parameter learning task is to automatically learn the weights w_i associated with the features F_i from data by optimizing a given objective function. Ideally, each candidate MLN is scored by its training set log-likelihood, which is a convex function of its weights. Hence, learning the optimal weights can be solved via convex optimization. However, parameter learning for MLNs poses the same challenges as parameter learning for MRFs that are discussed in Section 2.2.3. Therefore, the standard approach for learning the weights of an MLN is optimizing the pseudo-log-likelihood as defined in Equation 2.3 (Huynh and Mooney 2009; Lowd and Domingos 2007; Richardson and Domingos 2006; Singla and Domingos 2005).

The observation that parameter learning quickly becomes intractable, motivated Van den Broeck, Meert, et al. (2013) to leverage lifted-inference techniques to address the parameter-learning task. Their lifted-parameter-learning approach efficiently and exactly learns the maximum-likelihood weights for a given MLN. The algorithm compiles the MLN into a more convenient circuit language that permits inference in polynomial time in the domain size of the MLN.

2.3.4 Structure Learning

The structure learning task is to automatically learn the formulas F_i as well as their corresponding weights w_i from data. Structure learning is an incredibly challenging task as there is a huge number of candidate formulas and an

even larger space of candidate models. The traditional approach to this task is to greedily add one formula at a time to the MLN. The existing approaches that address this task can be largely divided into two categories: top-down approaches and bottom-up approaches.

Top-down Approaches

The first category of structure learners adopts a top-down approach. The MSL algorithm by Kok and Domingos (2005) is a canonical example of a top-down approach. Starting from an MLN that only contains the unit formulas, MSL proceeds as follows. After constructing all formulas of length two, it runs a beam search to find the current best formula and adds that formula to the model. In each iteration, MSL constructs new candidate formulas by adding literals to the best formulas in the beam. The search iterates until no formula improves the score of the MLN.

To evaluate the merit of each formula, MSL uses weighted pseudo-log-likelihood (WPLL), which is an extension of pseudo-log-likelihood that diminishes the importance of predicates with a large number of ground instances. It does this by normalizing a predicate's pseudo-log-likelihood by its number of possible ground instances (Kok and Domingos 2005). To avoid overfitting, each formula receives a penalty term proportional to the number of literals that differ between the current formula and the initial formula.

Bottom-up Approaches

The second category of structure learners adopts a bottom-up approach. Structure learners in this category, such as the BUSL algorithm by Mihalkova and Mooney (2007) and the LSM algorithm by Kok and Domingos (2010), use the data to restrict the search space.

The BUSL algorithm proceeds in two steps. First, it constructs a template Markov random field from a ground Markov random field by discovering recurring paths of true atoms. Second, it transforms the template Markov random field into candidate formulas. It greedily iterates through the set of candidate formulas adding the formula to the MLN that most improves the weighted pseudo-log-likelihood of the model. The search terminates when no formula improves the weighted pseudo-log-likelihood of the model.

2.4 Transfer Learning

Transfer learning is a subfield of machine learning that aims to learn predictive models leveraging data from the domain at hand as well as another, possibly unrelated, domain. This section introduces the relevant aspects of deep-transfer learning and reviews existing approaches. Pan and Yang (2010) provides a more thorough discussion of transfer-learning approaches and algorithms.

2.4.1 Definition

Transfer learning is a broad field and lacks a widely-adopted formal definition. Therefore, we formally define transfer learning as follows:

Given: A target task T, a target domain D_t and a source domain D_s, where the source domain D_s differs from the target domain D_t.

Learn: A target predictive function f_t in D_t using knowledge acquired in D_s.

Hence, the key difference between inductive learning and transfer learning is that the latter approach leverages additional data from a source domain to learn a more accurate predictive function.

2.4.2 Approaches

The work in this dissertation belongs to the class of deep-transfer-learning methods, which are capable of generalizing knowledge between distinct domains. DTM (Davis and Domingos 2009) and TAMAR (Mihalkova, Huynh, et al. 2007) are two state-of-the-art methods that perform deep-transfer learning in the context of Markov logic networks.

Furthermore, the field of analogical reasoning (Falkenhainer et al. 1989) is closely related to transfer learning. Analogical-reasoning approaches apply knowledge from one domain to another via a mapping between the objects and relations in the two domains. A human usually needs to provide possible mappings for each pair of domains, which can be tedious and time consuming.

DTM

DTM transfers knowledge from one domain to another using second-order cliques, which are sets of literals with predicate variables representing a set of formulas. For example, the clique $\{R(X,Y), R(Y,X)\}$, where R is a predicate variable and X and Y are object variables, gives rise to the second-order formulas $R(X,Y) \wedge R(Y,X)$, $R(X,Y) \wedge \neg R(Y,X)$, $\neg R(X,Y) \wedge R(Y,X)$, and $\neg R(X,Y) \wedge \neg R(Y,X)$. In turn, each of these second-order formulas gives rise to one or multiple first-order formulas.

DTM uses the data in the source domain to evaluate a set of second-order cliques and transfers a user-defined number of these cliques to the target domain. In the target domain, DTM first considers models only involving the formulas from the transferred cliques and then refines the models to tailor them more to the data in the target domain.

TAMAR

TAMAR first takes a first-order model for the source domain and then attempts to map each clause in that model to the target domain. The algorithm replaces the predicate symbols in each of the clauses with predicates from the target domain in all possible ways. TAMAR is less scalable than DTM in certain scenarios. In particular, exhaustively mapping each source clause to the target domain is time consuming for long clauses or clauses with constants if the target domain has many predicates or constants.

3

Structure Learning of Markov Random Fields

Markov random fields are undirected graphical models for compactly representing joint probability distributions over sets of random variables. The objective of structure learning is to automatically discover conditional dependencies and independencies in the data in order to represent the joint probability distributions more compactly. Markov random fields are often more conveniently represented as log-linear models, where the structure-learning task resorts to a feature-induction problem.

The traditional approach to structure learning is through standard search-based techniques. Algorithms that follow this strategy use the current feature set to construct a set of candidate features. After evaluating each feature, the highest-scoring feature is added to the model. The search can follow a top-down, general-to-specific strategy (e.g., Della Pietra et al. 1997; McCallum 2003) or bottom-up, specific-to-general strategy (e.g., Davis and Domingos 2010; Mihalkova and Mooney 2007). Search-based approaches tend to be slow due the large number of candidate structures that they need to explore. Furthermore, scoring each candidate structure requires learning the weights of the features through iterative optimization. Unfortunately, each iteration of weight learning requires running inference in the model, which is often intractable.

An alternative approach involves first learning a set of local models and then

combining them into a global model. Algorithms that follow this strategy consider each variable in turn and build a model to predict the value of a variable given the values of the remaining variables. Each predictive model is transformed into a set of features, which are included in the final, global model. Two approaches that use this strategy are the algorithm by Ravikumar et al. (2010), which employs L_1 logistic regression as the local model, and DTSL by Lowd and Davis (2010), which uses a probabilistic decision-tree learner as the local model. Unfortunately, learning the local models can be computationally expensive in domains containing a large number of variables or examples.

Contributions of this Chapter

The first contribution of this chapter is a novel Markov-random-field structure-learning algorithm called GSSL, which combines some of the benefits of recent approaches to structure learning. The proposed algorithm proceeds in two steps. The first step involves quickly generating a large set of candidate features by combining aspects from randomization and specific-to-general search. GSSL proceeds in a data-driven, bottom-up fashion to explore the space of candidate features and thus only constructs features that have support in the data. The second step selects a subset of the features to include in the final model. GSSL follows the philosophy of local-model-based approaches that try to minimize the computational expense of weight learning by performing weight learning only once to select the best features.

The second contribution is a large-scale empirical evaluation on 20 real-world datasets, which demonstrates the advantages of the proposed algorithm. Despite its simplicity, GSSL learns more accurate models than the baseline approaches while being much faster. The empirical evaluation introduces seven additional datasets, which have commonly been used as a benchmark for evaluating Markov-random-field structure-learning algorithms in recent years.

The Java source code of the GSSL implementation as well as a manual and tutorial are available on `https://dtai.cs.kuleuven.be/software/gssl`.

The content of this chapter is based on the following publication:

Jan Van Haaren and Jesse Davis (2012). "Markov Network Structure Learning: A Randomized Feature Generation Approach". In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012; Toronto, Ontario, Canada; 22-26 July 2012)*, pages 1148–1154

Structure of this Chapter

Section 3.1 introduces the key elements of the proposed GSSL algorithm. **Section 3.2** presents an extensive empirical evaluation on 20 real-world datasets that compares GSSL to three baseline approaches. **Section 3.3** provides conclusions and directions for future work.

3.1 Algorithm

We now describe GSSL (**G**enerate **S**elect **S**tucture **L**earning), an algorithm for Markov-random-field structure learning. GSSL has two main steps: feature generation and feature selection. In the feature generation step, starting from an initial feature set, GSSL quickly generates a large set of candidate features by combining aspects from randomization and specific-to-general search. In the feature selection step, GSSL attempts to discard irrelevant features through a preprocessing step and by applying weight learning with an L_1-penalty.

The four key elements of GSSL, which are introduced in the next subsections, are how to construct the initial feature set, how to generate new features, how to perform feature selection and how the overall algorithm functions.

3.1.1 Initial Feature Set Construction

The algorithm requires an initial feature set. Since the generation process generalizes features, the initial features should be specific so that it is possible to generalize them. The training examples can provide very specific features and GSSL considers two ways of converting them into the initial feature set. The first approach creates one long feature for each unique training example by forming a conjunction over all variables in that example. Because the features are maximally specific, generalization can generate every feature that has support in the data. The second approach works for problems that have only binary variables and it builds "positive" features by forming a conjunction only over those variables that have a value of true. Because many domains are sparse, this conversion has the advantage of being more compact. Both approaches have the advantage that every initial feature has support (i.e., occur) in the data. Consequently, generalizing any of these features yields a feature that is guaranteed to match at least one training example.

#	V_0	V_1	V_2	V_3	V_4
1	true	false	false	true	true
2	true	false	true	false	true
3	false	true	true	true	true
4	true	false	true	false	true

Table 3.1: An example dataset for Markov-random-field structure learning.

The example dataset in Table 3.1 illustrates the conversion process. Each row in the table corresponds to one training example.

The first approach yields the following initial feature set:

1a $V_0 = 1 \wedge V_1 = 0 \wedge V_2 = 0 \wedge V_3 = 1 \wedge V_4 = 1$

2a $V_0 = 1 \wedge V_1 = 0 \wedge V_2 = 1 \wedge V_3 = 0 \wedge V_4 = 1$

3a $V_0 = 0 \wedge V_1 = 1 \wedge V_2 = 1 \wedge V_3 = 1 \wedge V_4 = 1$

The second approach yields the following initial feature set:

1b $V_0 = 1 \wedge V_3 = 1 \wedge V_4 = 1$

2b $V_0 = 1 \wedge V_2 = 1 \wedge V_4 = 1$

3b $V_1 = 1 \wedge V_2 = 1 \wedge V_3 = 1 \wedge V_4 = 1$

In this example, the initial feature set is smaller than the example dataset since GSSL removes duplicate training examples as a preprocessing step. The second feature matches both the second and the fourth training example.

3.1.2 Feature Generation

The key step involves generating the features. To create a new feature, GSSL uniformly picks a feature, f, at random from the feature set. It generalizes f by dropping n arbitrary variables, where n is drawn from the uniform distribution between 1 and $l - 2$, with l the length of f. The lowerbound ensures that the new feature is actually a generalization. The upperbound ensures the length of

the generalized feature is at least length two since atomic features (i.e., features of length one) cannot be further generalized. To decide which variables to drop from the feature, GSSL shuffles the order of the variables in the ungeneralized feature and drops the first n to create a new feature $f^{'}$. Since $f^{'}$ is added back into the feature set, it can be selected for generalization in the future. Although this does not change which features can be generated, it biases the generation towards shorter features. This process is repeated for a fixed number of times and the same generalization can be produced multiple times. As a final step, GSSL adds an atomic feature for each variable, which captures its marginal probability, to its feature set. The atomic features are known to improve the performance in practice as they allow the conjunctive features to focus on the interactions between the variables.

To illustrate the process, suppose that GSSL picks feature **1a** from the dataset shown in Table 3.1. Since this feature is of length five, it uniformly draws an arbitrary number between one and three. Let us assume that this number is two. The algorithm then continues with dropping two arbitrary variables, say V_2 and V_3, such that the resulting feature is $V_0 = 1 \wedge V_1 = 0 \wedge V_4 = 1$. The new feature is more general and now also matches the second and fourth training example in addition to the first training example.

3.1.3 Feature Selection

GSSL does feature selection in two steps. In the first step, GSSL tries to identify and discard unnecessary features. One idea would be to perform pruning based on the support of each feature (i.e., how many examples it matches) in the data. However, GSSL does not count the number of training examples that each feature matches as this is a computationally expensive endeavor. As a result, GSSL requires another mechanism to identify features that potentially have high support in the data. GSSL removes any feature that was generated fewer times than a given threshold value. The use of a threshold is based on the assumption that GSSL is likely to generate features with high support in the data more often. In a second step, the algorithm performs L_1 weight learning on the remaining features to produce the final model. Placing a L_1-penalty on the magnitude of the weight forces many of the weights to be zero, which has the effect of removing them from the model. Hence, the weight learning process helps select the most relevant features.

The feature-selection approach is similar to that of Huynh and Mooney (2008) for discriminative structure learning for Markov logic networks, which are templates for constructing Markov random fields. Their approach first generates features in the form of first-order definite clauses and then uses weight learning with L_1-regularization to select a subset of the features.

3.1.4 Algorithm Overview

The overall control structure of GSSL proceeds in two phases. The first step, outlined in Algorithm 1, generates a large number of features. As input, this subroutine receives a set of training examples, TS, and the desired number of non-unique features to be generated, max. The set TS is converted into the initial feature set FS. Then, a feature f is randomly selected, generalized to f', and f' is added to FS. The procedure iterates until it has generated max number of non-unique features. Finally, it adds an atomic feature for each variable to FS.

Algorithm 1: GSSL — Feature Generation

Input: TS — set of training examples
max — maximum number of non-unique features
Output: FS — set of features

```
1  Feature set FS ← Convert TS into features
2  while |FS| < max do
3  |   Uniformly draw a feature f from FS
4  |   Let l be the length of f
5  |   Uniformly draw a number n ∼ [1, l − 2]
6  |   Drop n arbitrary variables from feature f
7  |   FS ← FS ∪ {f}
8  end
9  Add unit clause for each variable to FS
10 return FS
```

The second step is outlined in Algorithm 2. As input, this subroutine receives the set of generated features, FS, and a lower bound, $thres$, on the number of times each feature was proposed in the feature-generation step. First, the algorithm loops through the feature set and discards those features that were generated fewer than $thres$ times. Second, the algorithm learns the weights for each feature through L_1-optimization, which reduces the number of features in the model by forcing many weights to be zero.

Algorithm 2: GSSL — Feature Selection

Input: FS — initial set of features
thres — lower bound on the desired number of occurrences
Output: FS — final set of features

```
1 foreach feature f in FS do
2 |   if f occurs ≤ thresh times then
3 |   |   FS ← FS \ {f}
4 |   end
5 end
6 Perform L1 weight learning on FS
7 return FS
```

3.2 Experimental Evaluation

In this section, we evaluate GSSL on 20 real-world datasets. The evaluation consists of two parts. In the first part, we compare GSSL to three state-of-the art Markov-random-field structure-learning algorithms in terms of accuracy and runtime. More specifically, we compare GSSL to the DTSL algorithm by Lowd and Davis (2010), the L1 algorithm by Ravikumar et al. (2010) and the BLM algorithm by Davis and Domingos (2010). In the second part, we investigate the effect of GSSL's parameter values on its performance.

3.2.1 Datasets

Table 3.2 describes the characteristics of each dataset. Note that each dataset only contains binary variables. The datasets are shown in ascending order by number of variables. We used the 13 datasets from Lowd and Davis (2010). Additionally, we used seven new datasets: Accidents,[1] Ad,[2] BBC,[3] DNA,[4] Kosarak,[1] Pumsb Star[1] and Retail.[1] For Ad and DNA, we used all the provided binary features. For BBC, we created one binary feature for each word in the training set. The remaining four datasets are for frequent itemset mining. We subsampled these datasets and divided our subsample into a training, a tuning, and a test set. We

[1] These datasets are available on http://fimi.ua.ac.be/data/.
[2] This dataset is available on http://archive.ics.uci.edu/ml/datasets.html.
[3] This dataset is available on http://mlg.ucd.ie/datasets/bbc.html.
[4] This dataset is available on http://www.cs.sfu.ca/~wangk/ucidata/dataset/DNA/.

Dataset	Size of train set	Size of tune set	Size of test set	Number of variables	Density
NLTCS	16,181	2,157	3,236	16	0.332
MSNBC	291,326	38,843	58,265	17	0.166
KDDCup 2000	180,092	19,907	34,955	64	0.008
Plants	17,412	2,321	3,482	69	0.180
Audio	15,000	2,000	3,000	100	0.199
Jester	9,000	1,000	4,116	100	0.608
Netflix	15,000	2,000	3,000	100	0.541
Accidents	12,758	1,700	2,551	111	0.291
Retail	22,041	2,938	4,408	135	0.024
Pumsb Star	12,262	1,635	2,452	163	0.270
DNA	1,600	400	1,186	180	0.253
Kosarak	33,375	4,450	6,675	190	0.020
MSWeb	29,441	3,270	5,000	294	0.010
Book	8,700	1,159	1,739	500	0.016
EachMovie	4,524	1,002	591	500	0.059
WebKB	2,803	558	838	839	0.064
Reuters-52	6,532	1,028	1,540	889	0.036
20 Newsgroups	11,293	3,764	3,764	910	0.049
BBC	1,670	225	330	1,058	0.078
Ad	2,461	327	491	1,556	0.008

Table 3.2: An overview of the characteristics of the datasets showing, for each dataset, the number of features in the train, tune, and test set as well as the number of variables and density.

counted each item's number of occurrences in the training set. We constructed one binary feature for each item that met a particular threshold on the training set: 500 for Accidents and Pumsb Star, and 50 for Kosarak and Retail.

3.2.2 Methodology

We used the training data to learn the structure and weights for all four methods. The code for GSSL is publicly available.[5] We used the publicly available code for DTSL and BLM to learn the models. For L1, we used the OWL-QN software package from Andrew and Gao (2007) for performing L_1 logistic regression.

[5]The implementation is available on https://dtai.cs.kuleuven.be/software/gssl.

For GSSL, we generated half a million, one million, two million and five million features and used pruning thresholds of one, two and five. We tried both methods for converting the training set to the initial feature set. For the baseline algorithms, we used the parameter settings described in Lowd and Davis (2010).

Since GSSL, DTSL and L1 all produce feature sets, it is necessary to learn the weights for each feature. For weight learning, we used the Libra Toolkit[6] to optimize the train set pseudo-likelihood, which was done for computational tractability. In order to allow for a fair comparison, we performed the exact same weight learning procedure and employed the same set of L_1 regularization parameters for all three algorithms. For each dataset, we used Gaussian priors with standard deviations of 0.1, 0.5 and 1, combined with L_1 norm weights of 1, 5 and 10, resulting in nine different setups. For each algorithm, we selected the model that maximized the pseudo-log-likelihood on the validation set.

We evaluated the best model using test set conditional marginal log-likelihood (CMLL) (Lee et al. 2007; Lowd and Davis 2010). First, we divided the variables into a query set Q and an evidence set E. Then, we computed $CMLL(X = x) = \sum_{i \in Q} logP(X_i = x_i|E)$ for each example in the test set. We divided the variables into four disjoint groups for each dataset. One set served as query variables while the remaining three sets served as evidence. We repeated this procedure such that each set served as the query variables once. We computed the conditional marginal probabilities using the Gibbs sampler that is part of the Libra Toolkit. We used a burn-in of 100 samples and then computed the probability using the next 1,000 samples.

3.2.3 Results

Table 3.3 reports the CMLLs, averaged over all test examples, for each of the algorithms on all 20 datasets. We compared GSSL with each of the baselines using a Wilcoxon signed-rank test, which is a non-parametric, paired difference test. The comparison between any two algorithms involves 20 paired samples, where each sample corresponds to the test set CMLL scores on a different dataset. GSSL achieves the best overall CMLL score on 8 of the 20 datasets. According to the Wilcoxon signed-rank test, GSSL is equivalently accurate to L1, where it achieves a better CMLL score on 11 of the 20 datasets. GSSL significantly outperforms DTSL at the 0.0193 significance level according to a Wilcoxon

[6]The Libra Toolkit is available on http://libra.cs.uoregon.edu.

Dataset	GSSL	L1	DTSL	BLM
NLTCS	**-5.175**	-5.232	-5.209	-5.248
MSNBC	-5.947	-6.281	**-5.727**	-5.815
KDDCup 2000	-2.071	-2.108	**-2.046**	-2.077
Plants	**-9.854**	-10.739	-10.709	-10.445
Audio	**-36.803**	-36.878	-37.484	-37.452
Jester	**-49.464**	-49.476	-50.252	-52.762
Netflix	**-52.339**	-52.401	-53.342	-56.521
Accidents	-18.180	**-16.543**	-16.957	-37.558
Retail	-10.547	**-10.534**	-10.578	-10.620
Pumsb Star	-17.245	**-13.905**	-19.508	-133.155
DNA	-81.034	**-69.035**	-69.197	-99.560
Kosarak	-10.137	-10.183	**-10.068**	-10.217
MSWeb	**-8.819**	-8.959	-16.201	-8.848
Book	-34.048	**-34.025**	-34.120	-34.650
EachMovie	**-49.873**	-50.002	-51.448	-58.582
WebKB	-144.206	**-143.290**	-148.192	-164.844
Reuters-52	-79.501	**-78.743**	-81.267	-90.852
20 Newsgroups	-148.565	**-147.007**	-151.723	-160.841
BBC	-242.424	**-239.642**	-250.302	-265.486
Ad	**-14.848**	-15.393	-16.751	-45.638

Table 3.3: An overview of the test-set CMLL scores averaged over all test examples for each method and each dataset. The best score for each dataset is shown in bold. GSSL achieves the best CMLL score on 8 of the 20 datasets.

signed-rank test, producing a better CMLL score on 15 of the 20 datasets. GSSL significantly outperforms BLM at the 0.0002 significance level according to a Wilcoxon signed-rank test, beating BLM on 19 of the 20 datasets. Despite its simplicity, GSSL is often significantly more accurate than its competitors.

GSSL performs remarkably worse than its closest competitor L1 in Pumsb Star and DNA. The characteristics of the domains are a likely explanation for this observation. Both domains combine a reasonably large number of variables with a reasonably high density, which leads to a large search space for our randomized approach. This result suggests that GSSL would likely achieve better performance in these domains for a larger number of generated features.

GSSL exhibits outstanding runtime performance, which is reported in Table 3.4. GSSL is the fastest algorithm on 13 of the 20 datasets, being slower to L1 or

	GSSL			L1			DTSL			BLM
Dataset	**FG**	**WL**	**Total**	**FG**	**WL**	**Total**	**FG**	**WL**	**Total**	**Total**
NLTCS	5	69	74	5	4	9	2	5	**7**	8,738
MSNBC	5	218	223	72	38	**110**	126	80	206	458,691
KDDCup 2000	7	114	**121**	624	21	645	780	75	855	2,814,585
Plants	8	375	383	194	38	232	34	84	**118**	128,825
Audio	9	186	**195**	166	126	292	49	153	202	184,858
Jester	10	192	202	321	193	514	27	108	**135**	107,156
Netflix	10	333	**343**	385	322	707	47	336	383	188,455
Accidents	9	1,171	1,180	833	116	949	43	242	**285**	166,767
Retail	8	165	**173**	398	100	498	112	126	238	414,684
Pumsb Star	9	891	900	939	219	1,158	43	116	**159**	164,378
DNA	9	38	47	53	25	78	6	14	**20**	20,614
Kosarak	9	259	**268**	1,113	171	1,284	238	214	452	881,393
MSWeb	8	222	**230**	2,634	186	2,820	485	303	788	1,773,963
Book	11	223	**234**	1,731	290	2,021	256	538	794	944,807
EachMovie	10	234	**244**	5,922	310	6,232	191	171	362	701,127
WebKB	11	186	**197**	7,652	718	8,370	212	167	379	775,808
Reuters-52	10	344	**354**	10,331	1,099	11,430	589	748	1,337	2,158,685
20 Newsgroups	11	663	**674**	19,615	2,375	21,990	1,455	690	2,145	5,279,550
BBC	11	136	**147**	6,028	555	6,583	173	87	260	625,467
Ad	9	131	**140**	8,711	208	8,919	740	216	956	2,677,445

Table 3.4: An overview of the runtimes in seconds for each method and each dataset, showing the feature-generation (FG) time, weight-learning (WL) time, and total runtime. The best runtime for each dataset is shown in bold. GSSL is the fastest method on 13 of the 20 datasets, being slower to L1 or DTSL only when datasets have very few variables.

Dataset	Generated features	Retained features	Average length
NLTCS	12,184	8,881	3.83
MSNBC	52,591	52,355	4.18
KDDCup 2000	102,666	21,023	3.34
Plants	52,709	52,499	3.24
Audio	12,991	11,317	2.20
Jester	9,679	9,461	1.99
Netflix	9,393	9,335	1.99
Accidents	40,669	39,236	3.28
Retail	19,617	8,813	2.51
Pumsb Star	26,419	24,337	3.16
DNA	39,088	25,936	2.99
Kosarak	97,023	24,193	2.85
MSWeb	57,576	24,234	2.90
Book	101,451	15,930	1.97
EachMovie	113,171	58,792	2.54
WebKB	179,472	43,787	2.01
Reuters-52	163,970	93,233	2.12
20 Newsgroups	159,012	88,005	2.01
BBC	215,000	43,566	1.97
Ad	78,797	33,222	2.31

Table 3.5: An overview of the statistics for the best GSSL model for each dataset, showing the number of generated features after thresholding, the number of features after weight learning, and the average feature length.

DTSL only when datasets have very few variables. GSSL is faster than L1 on 16 of the 20 datasets. On average, it exhibits a runtime that is 15 times faster than L1. GSSL is faster than DTSL in addition to being significantly more accurate. GSSL's runtime is lower than DTSL's on 13 of the 20 datasets and is twice as fast on average. Naturally, GSSL is significantly faster than BLM, showing an average speed-up of 4634, as it avoids the computational cost associated with running weight learning to evaluate each candidate feature.

Tables 3.5, 3.6, 3.7, and 3.8 show statistics for the best learned model on each dataset for each algorithm. The GSSL and L1 models have more features. L1 has the lowest average feature length because it is restricted to atomic and pairwise features. On average, GSSL results in shorter features than either DTSL or BLM.

Dataset	Generated features	Retained features	Average length
NLTCS	134	134	1.88
MSNBC	153	152	1.88
KDDCup 2000	2,080	1,433	1.95
Plants	2,404	2,286	1.96
Audio	5,049	4,878	1.97
Jester	5,008	4,966	1.97
Netflix	4,985	4,952	1.97
Accidents	5,840	5,786	1.98
Retail	9,113	3,383	1.96
Pumsb Star	6,475	6,392	1.97
DNA	4,302	4,167	1.96
Kosarak	7,771	4,725	1.95
MSWeb	33,828	11,548	1.97
Book	120,833	10,647	1.95
EachMovie	70,568	15,900	1.96
WebKB	216,123	35,901	1.97
Reuters-52	172,730	91,373	1.99
20 Newsgroups	193,177	120,881	1.99
BBC	270,623	42,297	1.97
Ad	212,663	23,786	1.93

Table 3.6: An overview of the statistics for the best L1 model for each dataset, showing the number of generated features, the number of features after weight learning, and the average feature length.

Apart from weight learning, GSSL relies on only two parameters: a desired number of features and a pruning threshold. Table 3.9 shows the parameters that yield the best model for GSSL as well as the effect of thresholding on the number of features. The number of features needed to get the best performance depends on the characteristics of the dataset. Generally, if a dataset has more variables or a higher density (i.e., a greater proportion of the variables that are true), it is necessary to generate more features. Intuitively, this makes sense. More variables increases the number of possible features such that we need to try more combinations to get the best feature set. A higher density suggests the presence of more regularities in the data such that the models will need more features to capture them. GSSL got the best results based on tune-set pseudo-log-likelihood with 0.5 million features three times, 1 million features

Dataset	Generated features	Retained features	Average length
NLTCS	2,958	2,185	6.15
MSNBC	24,530	21,435	10.33
KDDCup 2000	8,585	6,039	7.64
Plants	12,289	6,243	6.50
Audio	4,946	4,805	3.08
Jester	4,796	4,751	3.70
Netflix	6,659	6,604	3.74
Accidents	10,194	5,390	6.85
Retail	4,439	3,944	5.26
Pumsb Star	4,666	4,434	5.24
DNA	2,246	2,221	3.11
Kosarak	8,724	6,402	5.37
MSWeb	14,788	11,911	18.17
Book	11,720	6,589	3.57
EachMovie	19,568	9,399	4.44
WebKB	17,939	10,633	3.43
Reuters-52	30,684	19,660	4.33
20 Newsgroups	21,008	18,915	2.69
BBC	4,417	4,131	1.91
Ad	13,708	11,335	3.34

Table 3.7: An overview of the statistics for the best DTSL model for each dataset, showing the number of generated features, the number of features after weight learning, and the average feature length.

four times, 2 million features seven times and 5 million features six times. Our experiments show that using a pruning threshold value of two is generally better than a threshold value of one. Furthermore, using a threshold value of five seems to be too strict and rules out too many potentially useful features.

On average, GSSL spends more than 95% of its time on weight learning, which is reported in Table 3.4. The time depends on both the number of generated features and the pruning threshold. Generally, generating smaller feature sets and using a higher pruning threshold yields the lowest runtimes for weight learning. Averaged across all datasets and pruning thresholds, GSSL spends 124.87 seconds on weight learning when generating 0.5 million features, 206.31 seconds when generating 1 million features, and 442.45 seconds when

Dataset	Generated features	Average length
NLTCS	385	4.17
MSNBC	4,213	4.69
KDDCup 2000	4,877	3.25
Plants	2,469	5.95
Audio	1,938	2.21
Jester	992	8.27
Netflix	1,140	5.82
Accidents	1,329	8.33
Retail	2,823	2.12
Pumsb Star	5,789	28.20
DNA	1,413	10.74
Kosarak	3,860	2.95
MSWeb	5,756	3.01
Book	6,077	1.97
EachMovie	2,561	4.10
WebKB	3,029	8.61
Reuters-52	5,907	11.33
20 Newsgroups	4,256	9.69
BBC	2,299	9.56
Ad	2,370	3.87

Table 3.8: An overview of the statistics for the best BLM model for each dataset, showing the number of generated features and the average feature length.

generating 2 million features. We have omitted the runtime for 5 million features as running weight learning using a pruning threshold is often intractable. Averaged across datasets and number of generated features, GSSL spends 414.83 seconds on weight learning for a pruning threshold of one, 235.22 seconds for a threshold of two, and 123.52 seconds for pruning of five.

The other choice that GSSL has is in terms of its set of initial features. Using an initial feature set of only "positive features" (i.e., features that are only conjunctions over true variables) is better on 14 of the 20 datasets. Generally, limiting the feature set to positive features yields much faster weight learning.

Dataset	Generated features	Unique features	Selected features	Prune rate	Threshold
NLTCS	500,000	143,364	12,184	91.50%	2
MSNBC	2,000,000	491,885	52,591	89.31%	1
KDDCup 2000	5,000,000	373,041	102,667	72.48%	2
Plants	1,000,000	690,390	52,709	92.37%	1
Audio	1,000,000	524,688	12,991	97.52%	2
Jester	500,000	369,000	9,679	97.38%	2
Netflix	500,000	382,760	9,393	97.55%	2
Accidents	2,000,000	1,462,685	40,669	97.22%	1
Retail	2,000,000	168,979	19,617	88.39%	2
Pumsb Star	2,000,000	1,565,510	26,419	98.31%	1
DNA	2,000,000	1,383,102	39,088	97.17%	1
Kosarak	1,000,000	288,936	97,023	66.42%	1
MSWeb	1,000,000	149,689	57,576	61.54%	1
Book	2,000,000	855,347	101,451	88.14%	1
EachMovie	5,000,000	2,200,237	113,172	94.86%	2
WebKB	5,000,000	2,738,782	179,473	93.45%	2
Reuters-52	5,000,000	2,514,634	163,971	93.48%	2
20 Newsgroups	5,000,000	3,037,720	159,013	94.77%	2
BBC	5,000,000	2,975,003	215,000	92.77%	2
Ad	2,000,000	578,322	78,797	86.37%	1

Table 3.9: An overview of the parameters and statistics for the best GSSL model for each dataset, showing the number of generated features, the number of unique features, the number of features left after thresholding, the percentage of features pruned by thresholding, and the threshold value.

3.3 Conclusions

This chapter investigated the task of learning Markov random fields from data and presented a novel algorithm called GSSL that addresses this task.

The proposed GSSL algorithm explores the search space in an efficient way by combining a data-driven feature-generation procedure with randomization. The data-driven search ensures that *only the relevant areas* of the potentially huge search space are explored. At the same time, the randomization ensures that *all relevant areas* of the search space can be explored. GSSL avoids the computational cost of building local models and needs to perform weight learning only once.

Our extensive empirical evaluation on 20 real-world datasets shows that GSSL offers outstanding performance in terms of both accuracy and runtime, while striking in its simplicity. GSSL is 15 times faster than the L1 approach by Ravikumar et al. (2010) while being equivalently accurate. GSSL is also faster and significantly more accurate than both DTSL and BLM.

In the future, it may be possible to improve GSSL's runtime further by exploring more sophisticated pruning strategies. Reducing the number of features considered during weight learning would greatly improve its efficiency.

4

Lifted Structure Learning of Markov Logic Networks

Markov logic networks are probabilistic graphical models that combine Markov random fields with first-order logic. Although Markov logic networks provide a powerful statistical-relational-learning framework, they pose a great challenge for inference and learning. Using traditional algorithms, these tasks reduce to inference and learning in densely-connected Markov random fields with millions of random variables. A new class of lifted-inference algorithms (Kersting 2012; Poole 2003), which exploit the abundant symmetries in relational representations to speed up probabilistic inference, address the intractability of reasoning. This chapter addresses the intractability of learning by leveraging lifted-inference techniques. The objective is to learn models that are tractable in the sense that they permit lifted inference.

Learning Markov logic networks from data consists of two tasks. The parameter-learning task is to learn the weights associated with the formulas in a given model. This task corresponds to learning the feature weights in a log-linear Markov random field. The structure-learning task is to learn the formulas themselves in addition to their associated weights. For both tasks, the data consist of a set of relational databases.

The success of lifted-inference algorithms raises three important questions for the learning task. The first question is whether leveraging lifted-inference

techniques in the parameter-learning task improves the quality of the learned models. The second question is whether lifted inference can be used to learn models that support certain types of queries efficiently. The third question is whether the use of lifted-inference techniques affects the learned models in terms of generative and predictive quality. This chapter addresses these questions for the generative learning task, where the objective is to learn models that maximize the probability of observing the data.

Contributions of this Chapter

The first contribution of this chapter is a lifted-structure-learning algorithm called LSL that learns tractable MLN models. Leveraging the lifted-parameter-learning algorithm called LWL introduced by Van den Broeck, Meert, et al. (2013) as a subroutine, LSL optimizes the exact training-set log-likelihood and learns models that are guaranteed to support certain types of queries efficiently. In contrast, traditional MLN structure-learning algorithms resort to optimizing an approximate metric such as the pseudo-log-likelihood, which often leads to highly intractable models. The proposed LSL algorithm follows in a long tradition of tractable structure-learning algorithms for probabilistic graphical models (e.g., Chechetka and Guestrin 2007), and is among the first tractable structure-learning algorithms for statistical-relational-learning formalisms.

The second contribution is an extensive empirical evaluation of the proposed LSL algorithm as well as the LWL algorithm introduced by Van den Broeck, Meert, et al. (2013) on three real-world datasets. The LWL algorithm learns models with better test-set log-likelihoods than the baseline approaches. When learning tractable models, the LSL algorithm outperforms existing structure-learning algorithms in terms of test-set log-likelihood as well as conditional log-likelihood and area under the precision-recall curve on prediction tasks. Moreover, the LSL algorithm even outperforms off-the-shelf intractable structure learners on prediction tasks. This surprising result suggests that tractable models are a powerful hypothesis space, which is sufficient for many standard learning problems.

The Java source code of the LSL implementation, which is part of the WFOMC package, is available on `https://dtai.cs.kuleuven.be/software/wfomc`.

The content of this chapter is based on the following publications:

Jan Van Haaren, Guy Van den Broeck, Wannes Meert, and Jesse Davis (2016). "Lifted Generative Learning of Markov Logic Networks". In: *Machine Learning* 103(1), pages 27–55

Jan Van Haaren, Guy Van den Broeck, Wannes Meert, and Jesse Davis (2014c). "Tractable Learning of Liftable Markov Logic Networks". In: *Proceedings of the ICML 2014 Workshop on Learning Tractable Probabilistic Models (LTPM 2014; Beijing, China; 26 June 2014)*

Structure of this Chapter

Section 4.1 first discusses possible approaches to learning tractable models and then introduces the LSL algorithm. **Section 4.2** presents an extensive empirical evaluation that evaluates both the LWL algorithm introduced by Van den Broeck, Meert, et al. (2013) and the proposed LSL algorithm. More specifically, the evaluation investigates how accurate the learned weights are, how accurate the learned models are, and how well the learned models perform on prediction tasks. **Section 4.3** briefly discusses relevant related work. **Section 4.4** provides conclusions and directions for future work.

4.1 Algorithm

In this section, we describe a general structure learning approach and present two learning algorithms that use lifted-inference techniques. To optimally benefit from the existing lifted inference algorithms, we need a structure learning approach that learns liftable models, which are models that are tractable in the sense that they permit lifted inference.

The ideal approach to learn a liftable model is to design a search space that only contains liftable models. This is complicated by the fact that we still lack a full characterization of which models are liftable. We know that models where each formula contains at most two distinct logical variables are always liftable but this class of models may be too restrictive. However, many models that contain more expressive formulas are also liftable. This process is complicated further by the fact that two formulas may be liftable when they are considered independently but may lead to a model that is not liftable when they are combined into a single model. Furthermore, even if a model is liftable, the

underlying circuit representation may be too big to fit in memory or too time-consuming to evaluate. Hence, it is difficult to design a suitable search space.

We propose three possible solutions to this problem. The first solution is to integrate a check into the search procedure that verifies whether each candidate model is liftable such that unliftable models are discarded from the search space. Furthermore, a bias can be inserted into the search to avoid liftable models that are too big to fit in memory or too complex to be evaluated in practice. The second solution is to design a search space where each candidate formula contains at most two distinct logical variables such that the learned model is always liftable. The third solution is to run an off-the-shelf structure learning algorithm, such as BUSL or MSL, using the default objective function of WPLL, and to perform parameter tuning in such a way that the final learned model is liftable. However, the parameter-tuning process can be tedious and time-consuming since we have little understanding of which models are liftable.

In the following, we first introduce our general lifted-structure-learning approach and then present algorithms for the first two solutions. The algorithms can be viewed as concrete instances of the general approach.

4.1.1 Approach

Algorithm 3 outlines our **L**ifted **S**tructure **L**earning (LSL) approach, which learns a model of MLN formulas and their associated weights given a set of candidate MLN formulas and a set of relational training databases. The approach optimizes the training-set log-likelihood by iteratively adding formulas to an initially empty model.

In each iteration, the LSL approach performs three steps. First, LSL builds a set of candidate models by adding each candidate formula to a distinct copy of the current model. Second, LSL learns the associated formula weights and computes the training-set log-likelihood for each candidate model. Third, LSL replaces the current model by the best candidate model in terms of training-set log-likelihood if that model yields a log-likelihood improvement over the current best model. The algorithm terminates when no more candidate formulas are available or none of the remaining candidate models yields a log-likelihood improvement over the current best model.

To compute the training-set log-likelihood for a candidate model, LSL employs an internal cross-validation approach. LSL learns the formula weights on all-but-

Algorithm 3: LSL — Lifted Structure Learning

Input: CFS — set of candidate MLN formulas
DB — set of relational training databases
Output: M — the learned MLN model

```
1  M ← ∅
2  M_LL ← 0
3  while |CFS| > 0 do
4      BCM ← ∅
5      BCF ← ∅
6      BCM_LL ← 0
7      forall the candidate formula CF ∈ CFS do
8          CM ← M ∪ CF
9          WCM ← LiftedWeightLearning(CM,DB)
10         WCM_LL ← ComputeLogLikelihood(WCM,DB)
11         if WCM_LL > BCM_LL then
12             BCM ← WCM
13             BCF ← CF
14             BCM_LL ← WCM_LL
15         end
16     end
17     M ← BCM
18     CFS ← CFS \ {BCF}
19 end
20 return M
```

one training database and computes the log-likelihood on the left-out database. LSL repeats this procedure such that each database served as the validation database once. To obtain the log-likelihood for a candidate model, LSL simply averages the log-likelihoods across the validation databases.

4.1.2 Algorithm I: Perform a Tractability Check

The first algorithm checks the tractability of candidate models by restricting the time spent on the second step (i.e., lines 9-10 in Algorithm 3), which is concerned with learning the formula weights and computing the training-set log-likelihood. Each candidate formula that cannot be compiled into a circuit or whose weight cannot be learned within the allotted time, is discarded. By

biasing the search process towards formulas that can be compiled into a circuit, the tractability of the final model is ensured.

The initial set of candidate formulas can be constructed either by running the candidate formula construction step of an off-the-shelf structure learning algorithm such as BUSL or MSL, or by greedily enumerating all formulas satisfying certain constraints. In our experiments, we enumerate all valid formulas having at most three literals and at most three distinct object variables. Furthermore, we only consider the subset of "connected" formulas for which a path via the arguments exists between any two literals.

4.1.3 Algorithm II: Design a Space of Tractable Models

The second algorithm gets around the tractability check by restricting the initial set of candidate formulas (i.e., input variable *CFS* in Algorithm 3) to formulas that meet certain constraints. As a result, we can easily exploit the observation that models consisting of formulas containing at most two distinct logical variables are always tractable. Specifically, we can design an appropriate search space by enumerating all valid formulas comprising at most two distinct object variables up till a specific number of literals.

4.2 Experimental Evaluation

In this section, we evaluate both the **L**ifted **W**eight **L**earning (LWL, Van den Broeck, Meert, et al. 2013) and **L**ifted **S**tructure **L**earning (LSL) approaches.[1] We first present the experimental setup and then address the following five research questions related to weight learning and structure learning.

- **Q1:** Does exactly optimizing the log-likelihood during weight learning with LWL lead to more accurate formula weights?
- **Q2:** How does LSL compare to the off-the-shelf structure learners in terms of log-likelihood when learning *tractable* models?
- **Q3:** How does LSL compare to the off-the-shelf structure learners in terms of AUC and CLL when learning *tractable* models?

[1] The implementations of both LWL and LSL are available on https://dtai.cs.kuleuven.be/software/wfomc as part of the WFOMC package.

- **Q4:** How does LSL compare to the off-the-shelf structure learners in terms of AUC and CLL when learning *intractable* models?
- **Q5:** What is the effect of the formula-evaluation time-out in LSL on the complexity and quality of the models for each of the algorithms?

4.2.1 Experimental Setup

This subsection introduce the datasets, explains how the different models are learned, and discusses the inference setup for the prediction tasks.

Datasets

In our experiments, we use the following three real-world datasets:

- The **IMDb** dataset comes from the IMDb.com website (Mihalkova and Mooney 2007). The dataset contains information about attributes (e.g., gender) and relationships among actors, directors, and movies. The dataset consists of five different databases or folds.
- The **UWCSE** dataset contains information about the University of Washington CSE Department (Richardson and Domingos 2006). The data contains information about students, professors and classes, and models relationships (e.g., teaching assistant and advisor) among these entities. The dataset consists of five databases; one database for each of the five different groups in the CSE Department.
- The **WebKB** dataset consists of Web pages from the computer science departments of four universities in the United States (Mihalkova and Mooney 2007). The dataset contains information about labels of pages (e.g., student and course). The dataset consists of four databases; one database for each of the four universities.

In all domains, we perform cross-validation by holding out one fold as test set and learning a model on the remaining folds. Each fold serves as test set once.

Models

We compare tractable models learned by LSL with both tractable and intractable models learned by the bottom-up structure learner BUSL (Mihalkova and Mooney 2007) and the top-down structure learner MSL (Kok and Domingos 2005). We learned the considered models as follows:

- **Tractable LSL models:** LSL is run with all valid "connected" MLN formulas containing up to three literals and three distinct object variables as the initial set of candidate formulas. LSL discards any candidate model for which the LWL subroutine fails to find weights within the allotted time limit of five minutes.
- **Tractable BUSL and MSL models:** To enforce tractability, both BUSL and MSL are run with the restriction of learning formulas that contain no more than four literals and three distinct object variables.
- **Intractable BUSL and MSL models:** BUSL and MSL are run with their default parameter settings, which allows them to learn formulas containing up to five literals and five distinct object variables.

Inference Setup

In each domain, we predict the marginal probabilities of each predicate given evidence about all other predicates. Since lifted inference approaches cannot efficiently handle arbitrary binary evidence (Van den Broeck and Darwiche 2013), we use MC-SAT, which is part of the Alchemy package, to compute the probabilities. We use a burn-in of 10,000 samples and compute the probabilities with the following 100,000 samples. We measure the area under the precision-recall curve (AUC) and the test-set conditional log-likelihood (CLL) for the predicate of interest. AUC is insensitive to the large number of true negatives in the datasets, whereas CLL measures the quality of the probability estimates.

In our evaluation, we report the number of wins, losses, and ties for each algorithm. Since AUC and CLL are both skew-dependent metrics and the skew of a predicate varies across different predicates and different databases, simply averaging AUCs and CLLs, as has commonly been done in the past, is incorrect and leads to misleading results (Boyd et al. 2012).

4.2.2 Research Questions

Q1: Does Exactly Optimizing the Log-Likelihood During Weight Learning Lead to More Accurate Formula Weights?

This question investigates whether exactly optimizing the log-likelihood yields better models than optimizing the approximated likelihood during weight learning. We use the tractable BUSL and MSL models to address this question. For each structure, we learn the weights with the following three algorithms:

- **PLL:** This approximate approach optimizes the pseudo-log-likelihood of the weights via the limited-memory BFGS algorithm (Liu and Nocedal 1989). We use the implementation that is available in the Alchemy package (Kok, Sumner, et al. 2010).
- **PSCG:** This discriminative weight-learning approach by Lowd and Domingos (2007) optimizes the log-likelihood of the data by making all the predicates query atoms and hence leaving the evidence set empty. We use the implementation that is available in the Alchemy package (Kok, Sumner, et al. 2010).
- **LWL:** This is the lifted weight learning approach by Van den Broeck, Meert, et al. (2013). The approach uses the WFOMC package (Van den Broeck, Taghipour, et al. 2011) to compute the gradient and the limited-memory BFGS algorithm to optimize the weights.

First, each weight learning algorithm learns the weights for the given structures using the same data that produced each structure. Second, we compute the test-set log-likelihood for each model. Since we use the WFOMC package to compute the test-set log-likelihoods, the only difference among the three algorithms is in how the formula weights are learned.

Table 4.1 reports the test-set log-likelihoods for all three methods in all three domains for the models learned by BUSL. The table shows that LWL consistently outperforms both PLL and PSCG. Table 4.2 reports the test-set log-likelihoods for all three methods in all three domains for the models learned by MSL. The table shows that LWL again consistently outperforms both PLL and PSCG. These empirical results confirm our hypothesis that exactly optimizing the training-set log-likelihood results in more accurate formula weights.

	IMDb			UWCSE			WebKB		
	PSCG	PLL	LWL	PSCG	PLL	LWL	PSCG	PLL	LWL
F1	-566	-548	**-378**	-1,774	-1,860	**-1,524**	-863	-858	**-778**
F2	-548	-689	**-390**	-601	-594	**-535**	-1,422	-1,422	**-1,331**
F3	-1,223	-1,157	**-851**	-1,415	-1,462	**-1,245**	-717	-717	**-702**
F4	-425	-415	**-285**	-2,781	-2,820	**-2,510**	-1,224	-1,224	**-1,052**
F5	-423	-413	**-267**	-2,634	-2,763	**-2,357**	N/A	N/A	N/A

Table 4.1: The test-set log-likelihoods for all the methods in all the domains for the models learned by BUSL. LWL consistently outperforms both PLL and PSCG for all 14 models. The best result for each fold in each domain is in bold. The N/A entries arise because the WebKB dataset only contains four folds.

	IMDb			UWCSE			WebKB		
	PSCG	PLL	LWL	PSCG	PLL	LWL	PSCG	PLL	LWL
F1	-558	-831	**-440**	-1,761	-1,705	**-1,469**	-869	-868	**-797**
F2	-561	-944	**-477**	-594	-574	**-509**	-1,426	-1,426	**-1,324**
F3	-1,336	-1,576	**-909**	-1,382	-1,358	**-1,198**	-711	-711	**-677**
F4	-442	-393	**-315**	-2,745	-2,758	**-2,449**	-1,207	-1,207	**-1,054**
F5	-443	-388	**-353**	-2,616	-2,582	**-2,254**	N/A	N/A	N/A

Table 4.2: The test-set log-likelihoods for all the methods in all the domains for the models learned by MSL. LWL consistently outperforms both PLL and PSCG for all 14 models. The best result for each fold in each domain is in bold. The N/A entries arise because the WebKB dataset only contains four folds.

Q2: How Does LSL Compare to the Off-the-Shelf Structure Learners in Terms of Log-Likelihood When Learning Tractable Models?

This question investigates whether models learned by LSL yield better test-set log-likelihoods than tractable models learned by the off-the-shelf structure learning algorithms. We compare the tractable LSL models to the tractable BUSL and MSL models to address this question. We use LWL to learn the weights for the BUSL and MSL models since this approach outperforms the traditional weight learning algorithms (see Q1).

Table 4.3 reports the test-set log-likelihoods for tractable models learned by

	IMDb			UWCSE			WebKB		
	PSCG	PLL	LSL	PSCG	PLL	LSL	PSCG	PLL	LSL
F1	-378	-440	**-274**	-1,524	-1,469	**-1,407**	-778	-797	**-777**
F2	-390	-477	**-311**	-535	**-509**	-543	-1,331	**-1,324**	-1,341
F3	-851	-909	**-737**	-1,245	-1,198	**-1,157**	-702	-677	**-662**
F4	-285	-315	**-222**	-2,510	-2,449	**-2,409**	-1,052	-1,054	**-1,049**
F5	-267	-353	**-220**	-2,357	-2,254	**-2,089**	N/A	N/A	N/A

Table 4.3: The test-set log-likelihoods for all the methods in all the domains when learning tractable models. LSL outperforms both BUSL and MSL in terms of test-set log-likelihood in 12 of the 14 settings, doing only marginally worse than MSL on the second fold of the UWCSE and WebKB dataset. The best result for each fold in each domain is in bold. The N/A entries arise because the WebKB dataset only contains four folds.

BUSL, MSL, and LSL. LSL outperforms both BUSL and MSL in terms of test-set log-likelihood in 12 of the 14 settings, doing only marginally worse than MSL on the second fold of the UWCSE and WebKB datasets. These experimental results show that there is no reason to prefer an off-the-shelf structure learner to our lifted-structure-learning approach for learning tractable models when optimizing the test-set log-likelihood.

Q3: How Does LSL Compare to the Off-the-Shelf Structure Learners in Terms of AUC and CLL When Learning Tractable Models?

This question investigates whether models learned by LSL are better at answering queries than tractable models learned by the off-the-shelf structure learning algorithms. We compare the tractable LSL models to the *tractable* BUSL and MSL models to address this question. We use LWL to learn the weights for the BUSL and MSL models since this approach outperforms the traditional weight learning algorithms (see Q1).

Table 4.4 reports the number of times LSL wins, loses, and ties against the off-the-shelf structure learners BUSL and MSL in terms of both AUC and CLL. In terms of AUC, LSL beats BUSL in 61 of the 95 settings, ties in 6 settings, and loses in 28 settings. In terms of CLL, LSL beats BUSL in 72 of the 95 settings and loses in 23 settings. In terms of AUC, LSL beats MSL in 50 of the 95 settings,

		IMDb			UWCSE			WebKB		
Baseline	**Metric**	W	L	T	W	L	T	W	L	T
BUSL	AUC	**20**	5	5	**14**	6	0	**27**	17	1
BUSL	CLL	**25**	5	0	**15**	5	0	**32**	13	0
MSL	AUC	10	8	**12**	**12**	8	0	**28**	16	1
MSL	CLL	**23**	7	0	**15**	5	0	**32**	13	0

Table 4.4: A comparison of AUC and CLL results for all the methods in all the domains when learning tractable models. In comparison to BUSL, LSL wins in 133 of the 190 settings, ties in 6 settings, and loses in 51 settings. In comparison to MSL, LSL wins in 120 of the 190 settings, ties in 13 settings, and loses in 57 settings. The most frequent result for each comparison is in bold.

ties in 13 settings, and loses in 32 settings. In terms of CLL, LSL beats MSL in 70 of the 95 settings and loses in 25 settings.

LSL consistently leads to better models, achieving more wins than both BUSL and MSL on both metrics. Although we relearned the weights for the BUSL and MSL models using LWL, during structure learning these algorithms initially optimize WPLL, which has a very similar objective to CLL, and thus should be to their advantage on that metric. This makes it surprising that LSL does particularly well at CLL compared to BUSL and MSL.

Q4: How Does LSL Compare to the Off-the-Shelf Structure Learners in Terms of AUC and CLL When Learning Intractable Models?

This question investigates whether models learned by LSL are better at answering queries than models learned by the off-the-shelf structure learning algorithms. We compare the tractable LSL models to the *intractable* BUSL and MSL models to address this question. We use PLL to relearn the weights for the BUSL and MSL models. We cannot use LWL to relearn the weights since these models cannot be compiled into a circuit representation.

Table 4.5 reports the number of times LSL wins, loses, and ties against the off-the-shelf structure learners BUSL and MSL in terms of both AUC and CLL. In terms of AUC, LSL beats BUSL in 50 of the 95 settings, ties in 12 settings, and loses in 33 settings. In terms of CLL, LSL beats BUSL in 65 of the 95 settings, ties in 1 setting, and loses in 29 settings. In terms of AUC, LSL beats MSL in 47

		IMDb			UWCSE			WebKB		
Baseline	**Metric**	W	L	T	W	L	T	W	L	T
BUSL	AUC	**12**	7	11	**12**	8	0	**26**	18	1
BUSL	CLL	**17**	12	1	**15**	5	0	**33**	12	0
MSL	AUC	7	11	**12**	9	**11**	0	**31**	14	0
MSL	CLL	**16**	14	0	**11**	9	0	**28**	17	0

Table 4.5: A comparison of AUC and CLL results for all the methods in all the domains when learning intractable models. In comparison to BUSL, LSL wins in 115 of the 190 settings, ties in 13 settings, and loses in 62 settings. In comparison to MSL, LSL wins in 102 of the 190 settings, ties in 12 settings, and loses in 76 settings. The most frequent result for each comparison is in bold.

of the 94 settings, ties in 12 settings, and loses in 36 settings. In terms of CLL, LSL beats MSL in 55 of the 95 settings and loses in 40 settings.

Surprisingly, BUSL and MSL do only slightly better at answering queries when they are no longer bound to learning tractable models. Their performance remains roughly the same. These results show that learning longer, more complex formulas does not necessarily lead to much better inference results. A possible explanation is that more complex models may fit the data better but also lead to more complicated inference tasks, which in turn leads to a decreased predictive performance.

Q5: What Is the Effect of the Formula-Evaluation Time-Out in LSL on the Complexity and Quality of the Models for Each of the Algorithms?

This question investigates whether a restriction on the formula evaluation time leads to simpler models. To answer this question, we modify the off-the-shelf structure learners such that they can only spend a specified amount of time to evaluate each candidate formula. We run all three structure learning algorithms with three different formula evaluation time-outs: one minute, two minutes, and five minutes. Keeping all other parameter values unchanged, we learn both tractable and intractable models with the off-the-shelf structure learners.

Table 4.6 reports the test-set log-likelihoods for the models learned by LSL for all three formula-evaluation time-outs. In most settings, the time-out has either a small impact or no impact at all on the test-set log-likelihood. The only

	IMDb			UWCSE			WebKB		
	1 m.	2 m.	5 m.	1 m.	2 m.	5 m.	1 m.	2 m.	5 m.
F1	-276	-275	**-274**	-1,454	-1,421	**-1,407**	**-777**	-794	**-777**
F2	**-309**	-310	-311	-594	-562	**-543**	**-1,341**	**-1,341**	**-1,341**
F3	-739	-739	**-737**	-1,209	-1,162	**-1,157**	-664	-663	**-662**
F4	**-222**	**-222**	**-222**	-2,434	**-2,406**	-2,409	-1,049	**-1,043**	-1,049
F5	**-219**	-220	-220	**-2,089**	-2,142	**-2,089**			

Table 4.6: The test-set log-likelihoods for LSL in all the domains for three different time-out values: one, two, and five minutes. In most settings, the time-out value has either a small impact or no impact at all on the test-set log-likelihood. The best result for each fold in each domain is in bold.

	IMDb	UWCSE	WebKB
LSL (5-minute time-out)	5.40 ± 1.14	10.20 ± 0.45	4.75 ± 1.50
BUSL tractable	1.40 ± 0.55	7.60 ± 3.21	3.25 ± 0.50
MSL tractable	3.00 ± 0.00	6.80 ± 0.45	4.00 ± 1.15
BUSL intractable	6.60 ± 3.36	18.60 ± 10.55	8.50 ± 4.20
MSL intractable	4.20 ± 0.84	4.60 ± 1.95	4.00 ± 1.15

Table 4.7: The average number of formulas in the models learned by each algorithm in each domain.

domain that seems to benefit from a longer run time is UWCSE, which is the most complicated domain in terms of number of predicates and facts. These results show that LSL is robust to the formula-evaluation time-out value and confirm our observation that most structures can be compiled into a circuit either relatively quickly or not at all. Hence, restricting the formula-evaluation time for LSL does not lead to simpler models.

Furthermore, the formula-evaluation time-out also does not have an impact on the complexity of the models learned by the off-the-shelf structure learners. Both BUSL and MSL learn the same structures for all three time-out values. Since these algorithms do not need to run inference to evaluate a candidate formula, they require only little time per formula evaluation. Hence, restricting the formula evaluation time for the off-the-shelf structure learners does not lead to simpler models either.

	IMDb	UWCSE	WebKB
LSL (5-minute time-out)	2.69 ± 0.14	2.69 ± 0.09	2.85 ± 0.17
BUSL tractable	3.00 ± 0.35	2.18 ± 0.25	2.35 ± 0.31
MSL tractable	2.67 ± 0.00	2.65 ± 0.07	2.53 ± 0.22
BUSL intractable	3.12 ± 0.51	2.71 ± 0.22	3.02 ± 0.37
MSL intractable	4.23 ± 0.33	3.27 ± 0.48	2.53 ± 0.22

Table 4.8: The average length of the formulas in the models learned by each algorithm in each domain.

Tables 4.7 and 4.8 report the average number of formulas in a learned model and the average formula length for all learning methods in each domain. We compute these metrics because they are indicative of model complexity and allow us to further explore if LSL has a bias towards simpler models. Both BUSL and MSL include a complexity penalty based on formula length. The results show that all approaches tend to learn similarly-sized models. The exception is that BUSL, in the intractable setting, learns more formulas than the other approaches, and its formulas tend to be slightly longer on average.

These experimental results provide some evidence that LSL does not offer better performance simply because it has a preference for simpler models. Instead, regularization by liftability and support for maximum-likelihood learning account for the performance of the lifted-structure-learning approach.

4.3 Related Work

Learning tractable probabilistic models, that is, models that always permit efficient inference for certain types of queries, is an emerging area of research. The largest body of work restricts the structure of the learned models (e.g., Chechetka and Guestrin 2007), which can generally be done in two ways.

The first way is only considering low-tree-width models (Chechetka and Guestrin 2007; Narasimhan and Bilmes 2004). The second way is simultaneously learning an alternative representation that permits efficient inference in addition to a Bayesian network or Markov random field. This alternative representation is often an arithmetic circuit of the model. The model is penalized by the cost of inference, which can be calculated based on well-defined properties

of the representation. By penalizing the circuit size of the associated model, it is possible to bias the learning algorithm towards models where efficient inference is possible (Lowd and Domingos 2008; Lowd and Rooshenas 2013).

Our work follows the latter approach as our structure-learning approach learns models that allow lifted inference. As a result, the learned models are guaranteed to support tractable inference for certain types of queries. While tractable statistical-relational languages have been investigated before (Domingos and Webb 2012), we believe our work is among the first to consider the problem of learning such tractable representations.

4.4 Conclusions

This chapter investigated the task of learning Markov logic networks from data and presented a novel algorithm called LSL that addresses this task. More specifically, this chapter focused on the generative learning task, where the goal is to maximize the probability of observing the data.

The proposed LSL algorithm learns tractable Markov logic networks in the sense that they permit lifted inference. Unlike the competing approaches that resort to optimizing the pseudo-log-likelihood, LSL optimizes the exact log-likelihood. As a result, LSL is able to take better-informed decisions during the search as to whether to include a certain candidate formula into the model.

Our extensive empirical evaluation on three real-world datasets shows that LSL learns models with better test-set likelihood than the competing approaches. Furthermore, LSL also outperforms the competing approaches in terms of log-likelihood and area under the precision-recall curve on prediction tasks. More surprisingly, the tractable models learned by LSL also achieve better performance than intractable models on prediction tasks. This observation provides some evidence that tractable models are a powerful hypothesis space that is sufficient for many standard problems.

In the future, it may be possible to improve LSL's performance further by employing a beam-search strategy that explores several candidate models simultaneously. Alternatively, the current greedy-search strategy can be sped up by discarding the lowest-ranked candidate formulas in each iteration.

5

Deep Transfer Learning in Relational Domains

Traditional machine-learning algorithms focus on the paradigm of *inductive learning*, where a learning algorithm tries to generalize from the available training data in order to accurately classify test data from the same distribution as the training data. Learning accurate predictive models in this way can be challenging if only a limited amount of training data are available. Unfortunately, this is often the case in important real-world problems, where training data can be time-consuming, expensive or even impossible to obtain at all.

The field of *transfer learning* aims to exploit the observation that humans cope with a lack of training data quite well by transferring knowledge and intuitions from one setting to another. For example, people who are native Spanish speakers typically have fewer problems learning Italian than those who are native in Mandarin Chinese. Unlike inductive learning, transfer learning leverages additional data from a related domain in addition to data from the target task. In transfer learning, the former domain is considered the source domain, while the latter domain is considered the target domain.

Contributions of this Chapter

The first contribution of this chapter is a theoretical *deep-transfer-learning* framework called TODTLER. While traditional transfer-learning approaches, which perform *shallow-transfer learning* (Banerjee et al. 2006; Baxter et al. 1995), are concerned with tasks from the same domain, TODTLER can perform knowledge transfer between tasks from completely different domains. Moreover, TODTLER offers a principled view of transfer learning, unlike existing approaches for deep-transfer learning that offer ad-hoc solutions.

The second contribution is a practical implementation of the TODTLER framework. Unfortunately, an exact implementation of the framework would likely be very expensive computationally. Therefore, this dissertation presents an approximation, which allows the framework to be useful in practice.

The third contribution is an extensive empirical evaluation on three real-world datasets, which compares the proposed TODTLER framework to two baseline approaches. The evaluation shows that our approximation of the TODTLER framework outperforms the state-of-the-art deep-transfer-learning algorithm as well as the state-of-the-art inductive-learning algorithm in terms of accuracy. In addition to learning more accurate models, the TODTLER approximation is also much faster than the existing deep-transfer-learning algorithms. Furthermore, the empirical evaluation also introduces a novel Twitter dataset.

The Java source code of the TODTLER implementation as well as the Twitter dataset are available on `https://dtai.cs.kuleuven.be/software/todtler`.

The content of this chapter is based on the following publication:

Jan Van Haaren, Andrey Kolobov, and Jesse Davis (2015). "TODTLER: Two-Order-Deep Transfer Learning". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015; Austin, Texas, United States; 25-30 January 2015)*, pages 3007–3015

The theoretical framework is largely the work of Andrey Kolobov, while the practical implementation of the framework and the empirical evaluation are the work of the author of this dissertation.

Structure of this Chapter

Section 5.1 provides the intuition behind the TODTLER framework. **Section 5.2** presents the theoretical framework and shows how it can be used to transfer knowledge from one domain to another. **Section 5.3** introduces an approximation to the TODTLER framework, which allows it to be used in practice. **Section 5.4** presents an extensive empirical evaluation on three real-world datasets that compares TODTLER to two baseline approaches. **Section 5.5** provides conclusions and directions for future work.

5.1 Intuition

At a high level, TODTLER views knowledge transfer as the process of learning a declarative bias in the source domain and transferring that bias to the target domain to improve the learning process. More specifically, we concentrate on automatically learning Markov logic networks from data. We treat an MLN as an instantiation of a set of second-order templates expressible in a language called SOLT. The likelihood of an MLN model is thus partly determined by the learner's prior distribution over the sets of these second-order templates.

The main insight of our work is that transferring knowledge amounts to acquiring a posterior over the sets of second-order templates by learning in the source domain and using this posterior when learning in the target domain. As an example, consider the concept of transitivity, which can be expressed as a second-order template $R(X,Y) \wedge R(Y,Z) \implies R(X,Z)$, where R is a predicate variable. Therefore, this template is not specific to any domain, although its instantiations, e.g., $Knows(X,Y) \wedge Knows(Y,Z) \implies Knows(X,Z)$, are. In our framework, if learning in the source domain reveals instantiations of the transitivity template to contribute to highly likely models, the learning process in the target domain will prefer models with transitive relations as well.

5.2 Theoretical Framework

This section introduces the TODTLER framework in a more formal way. Suppose we have two datasets, one characterizing the smoking habits of a group of people and the other describing connections between terrorists.

An MLN learned on the smokers dataset may contain the first-order clause $Smokes(X) \wedge Friends(X,Y) \implies Smokes(Y)$, which captures the regularity that a friend of a smoker may likely be a smoker. A similar regularity may appear in the terrorism dataset: a person in the same organization with a terrorist may likely be a terrorist. We aim to generalize such regularities from one model to another but simply transferring first-order clauses does not help because the datasets are described by different relationships and properties.

What the domains have in common is the concept of multirelational transitivity described by the second-order clause

$$R_1(X) \wedge R_2(X,Y) \implies R_1(Y) \tag{5.1}$$

where R_1 and R_2 are predicate variables. It are these types of important structural patterns that TODTLER attempts to identify in the source domain and transfer to other domains. More specifically, the knowledge transfer occurs by biasing the learner in the target domain to favor models containing previously discovered regularities in the source domain.

5.2.1 Generative Model for the Data

TODTLER views data in any domain as being generated by the hierarchical process shown in Figure 5.1. The process starts by producing a second-order model of the data, which is denoted as $M^{(2)}$. Formally, $M^{(2)}$ is a set of second-order templates, where each template is a clause from a special language called SOLT (**S**econd-**O**rder **L**anguage for **T**emplates). SOLT is a restriction of second-order logic that allows only predicate variables and restricts the length of the clause. Equation 5.1 provides an example of a SOLT template. The power of SOLT stems from its ability to use predicate variables in order to state rules that reason about relations and thus describe domain-independent knowledge. The second-order model $M^{(2)}$ is sampled from a prior $P(M^{(2)})$ induced by independently including each template T expressible in SOLT into $M^{(2)}$ with some probability p_T. In particular, letting $p_T = 0.5$ for every template $T \in$ SOLT results in a uniform prior over all possible second-order models.

Given a second-order model $M^{(2)}$, a first-order MLN model $M^{(1)}$ is generated by instantiating all templates in $M^{(2)}$ with the set of predicates relevant to the

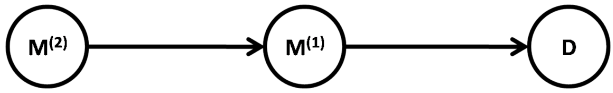


Figure 5.1: The data generation process.

data at hand in all possible ways. Instantiating a template with predicates means grounding each predicate variable in the template with a first-order predicate. In the earlier example, it is at this stage that the template in Equation 5.1 gives rise to the first-order formula $\text{Smokes}(X) \wedge \text{Friends}(X, Y) \implies \text{Smokes}(Y)$ as well as to many others. The weights for the first-order clauses produced in this way, which are necessary for a well-defined MLN, are sampled from a prior probability density, which is omitted in Figure 5.1. To complete the process, data is generated from the MLN using a relevant set of object constants to ground the first-order clauses. In the case of the smokers dataset, the data could include the ground facts $\text{Friends}(\text{Alice}, \text{Bob})$, $\text{Smokes}(\text{Alice})$, and $\text{Smokes}(\text{Bob})$.

Thus, letting a random variable D denote the data, the hierarchical generative model above yields a joint probability density $p(D, M^{(1)}, M^{(2)})$ factorizing as

$$p(D, M^{(1)}, M^{(2)}) = p(D|M^{(1)})p(M^{(1)}|M^{(2)})P(M^{(2)}). \tag{5.2}$$

In this formula, $p(D|M^{(1)})$ is given by Equation 2.4, $P(M^{(2)})$ is given by the probabilities p_T of including each template T into the second-order model, and $p(M^{(1)}|M^{(2)})$ is positive if the set of clauses in $M^{(1)}$ is the complete instantiation of $M^{(2)}$ and 0 otherwise. The set of clauses of an MLN $M^{(1)}$ is the complete instantiation of a second-order model $M^{(2)}$ if this set contains all first-order instantiations of all templates in $M^{(2)}$, with some formulas possibly having zero weights, and no other formulas.

For the cases when $p(M^{(1)}|M^{(2)}) > 0$, we can write the joint density $p(D, M^{(1)}, M^{(2)})$ in the following form, which is derived from Equation 2.4:

$$p(D, M^{(1)}, M^{(2)}) = \prod_{\vec{x} \in D} \left[\frac{1}{Z'} \prod_{T \in \mathcal{T}} p_T \exp\left(\sum_{F \in \mathcal{F}_T} w_F n_F(\vec{x}) \right) \right], \tag{5.3}$$

where p_T is the probability of including template T into second-order model $M^{(2)}$, $\mathscr{F}_T$ is the set of all first-order instantiations of the template T, and $\mathscr{T}$ is the set of all second-order templates expressible in SOLT. Equation 5.3 differs from Equation 2.4 in only one crucial aspect: the set of all first-order formulas is now divided into disjoint subsets of formulas corresponding to particular templates, where each subset has an associated probability p_T of being part of $M^{(1)}$. The probabilities p_T are the key means by which TODTLER transfers knowledge from one domain to another, as we explain next.

Algorithm 4: TODTLER — Framework

Input: D_s — source dataset
D_t — target dataset
$P(M^{(2)})$ — prior over second-order models

Output: $M_t^{(1)^*}$ — an MLN for the target domain

1 1. Find the posterior distribution $P_s(M^{(2)}|D_s)$ over second-order models such that $P_s(M^{(2)}|D_s)$ is encoded by the set of template probabilities $p_{T,s}$, given the data in the source domain and a similarly encoded prior $P(M^{(2)})$ over second-order models:

2

$$P_s(M^{(2)}|D_s) \leftarrow \frac{\int_{M_s^{(1)}} p(D_s, M_s^{(1)}|M^{(2)})P(M^{(2)})}{P(D_s)} \tag{5.4}$$

3 2. Determine the first-order MLN model that maximizes the joint probability of the data and the first-order model in the target domain if $P_s(M^{(2)}|D_s)$ is used as a prior over second-order models:

4

$$P_t(M^{(2)}) \leftarrow P_s(M^{(2)}|D_s) \tag{5.5}$$

$$M_t^{(1)^*} \leftarrow \arg\max_{M^{(1)}} p(D_t|M_t^{(1)}) \sum_{M^{(2)}} p(M_t^{(1)}|M^{(2)})P_t(M^{(2)}) \tag{5.6}$$

5.2.2 Transfer Learning with TODTLER

The TODTLER procedure is briefly summarized in Algorithm 4. Let D_s, $M_s^{(1)}$, D_t, and $M_t^{(1)}$ stand for the data and first-order MLN in the source and target domain, respectively. TODTLER performs transfer learning in two steps:

(1) Learning the Second-Order Model Posterior

In the first step, TODTLER finds the distribution $P_s(M^{(2)}|D_s)$ over second-order models that results from observing the data in the source domain given some initial belief $P(M^{(2)})$ over second-order models (Equation 5.4 in Algorithm 4). In view of Equation 5.3, determining $P_s(M^{(2)}|D_s)$ amounts to computing the template probabilities $p_{T,s}$ according to the source domain data and the prior.

(2) Target-Domain Learning Using the Posterior from the Source Domain

In the second step, TODTLER determines an MLN $M_t^{(1)^*}$ that maximizes the joint probability of the data and the first-order model in the target domain if the posterior $P_s(M^{(2)}|D_s)$ learned in the first step from the source data is used as the prior over second-order models (Equation 5.6 in Algorithm 4). As a result, TODTLER biases model selection for the target domain by explicitly transferring the experience of the learner from the source domain.

5.3 Approximate Algorithm

Despite the conciseness of TODTLER's procedural description (Algorithm 4), implementing it is non-trivial for several reasons. In this section, we discuss the challenges involved and present a series of appropriate approximations to the basic TODTLER framework. Algorithm 5 presents a pseudocode of the resulting implementation, which we will refer to throughout this section.

Algorithm 5: TODTLER — Approximation

Input: D_s — source dataset
D_t — target dataset
L — maximum template length
V — maximum number of distinct object variables
$P(M^{(2)}) = \{p_T^0\}_{T \in SOLT}$ — prior over second-order models

Output: $M_t^{(1)^*}$ — an MLN for the target domain

$\mathscr{T} \leftarrow$ EnumerateSecondOrderTemplates(L,V)
$\mathscr{F}_S \leftarrow$ EnumerateFirstOrderFormulas($\mathscr{T}$,D_s)
$\mathscr{F}_T \leftarrow$ EnumerateFirstOrderFormulas($\mathscr{T}$,D_t)
foreach second-order template $T \in \mathscr{T}$ **do**
  foreach first-order formula $F_{T,s} \in \mathscr{F}_S$ **do**
    $l_{F,s} \leftarrow$ WPLL of optimal $F_{T,s}$-MLN with respect to D_s
    $s_{F,s} \leftarrow l_{F,s}$ rescaled to $[0,1]$
  end
  $\hat{p}_{T,s} \leftarrow p_T^0 \cdot average_{F_{T,s} \in \mathscr{F}_S}\{s_{F,s}\}$
  foreach first-order formula $F_{T,t} \in \mathscr{F}_T$ **do**
    $l_{F,t} \leftarrow$ WPLL of optimal $F_{T,t}$-MLN with respect to D_t
    $s_{F,t} \leftarrow l_{F,t}$ rescaled to $[0,1]$
    $\hat{p}_{F,t} \leftarrow s_{F,t} \cdot \hat{p}_{T,s}$
  end
end
$M_t^{(1)^*} \leftarrow \{\}$
$previous\_WPLL \leftarrow -\infty$
$\mathscr{O}_F \leftarrow$ list of all $F_{T,t} \in \bigcup_{T \in \mathscr{T}} \mathscr{F}_T$ in decreasing order of $\hat{p}_{F,t}$
foreach first-order formula $F_{T,t} \in \mathscr{O}_F$ **do**
  $M_t^{(1)} \leftarrow M_t^{(1)^*} \cup \{F_{T,t}\}$
  $M_t^{(1)} \leftarrow$ relearn weights
  if WPLL of $M_t^{(1)}$ with respect to $D_t > previous\_WPLL$ **then**
    $previous\_WPLL \leftarrow$ WPLL of $M_t^{(1)^*}$ with respect to D_t
    $M_t^{(1)^*} \leftarrow M_t^{(1)}$
  end
end
return $M_t^{(1)^*}$

5.3.1 Learning Second-Order Model Posteriors

The main difficulty presented by TODTLER is computing the posterior distribution $P_s(M^{(2)}|D_s)$ over second-order models. Since we do not assume the distributions in Equation 5.4 to have any specific convenient form, it is not immediately obvious how to efficiently update the prior $P(M^{(2)})$. Moreover, Equation 5.4 involves summing over first-order MLNs, suggesting that an exact update procedure would likely be very expensive computationally.

Instead, we take a more heuristic approach. Our procedure exhaustively enumerates all SOLT templates that can form a user-specified maximum clause length L and maximum number of distinct object variables V (line 1). These conditions ensure that the number of second-order templates under consideration is finite and amount to adopting a prior $P(M^{(2)})$ that assigns probability 0 to any second-order model containing templates that violate these restrictions. Additionally, we assume that for each template T, its probability of inclusion $p_{T,s}$ under $P_s(M^{(2)}|D_s)$ is correlated with the "usefulness" of the first-order instantiations of T for modeling the data in the source domain and with its prior probability p_T^0.

For each first-order instantiation $F_{T,s}$ in the finite set $\mathscr{F}_S$ of all such instantiations of T generated by replacing T's predicate variables with predicates from the source domain, we calculate F's *usefulness score*, aggregate these numbers across $\mathscr{F}_S$, and use the result, along with the prior p_T^0, as a proxy $\hat{p}_{T,s}$ of $p_{T,s}$. The notion of usefulness of a single first-order formula is fairly crude — each formula typically contributes to the model along with many others, and its effect on the model's performance cannot be easily teased apart from that of the rest of the model. Nonetheless, as we explain next, this notion's simplicity also has a big advantage: a formula's usefulness can be computed very efficiently.

More specifically, for each $F_{T,s} \in \mathscr{F}_S$, we perform weight learning in the MLN that contains only the formula $F_{T,s}$. This MLN is denoted as the $F_{T,s}$-MLN. The weight learning process makes its own approximations as well. It optimizes the weights so as to maximize the weighted pseudo-log-likelihood (WPLL) of the model, which is an approximation to optimizing the log-likelihood. That, and the fact that our MLN contains only one formula, makes weight learning very fast. When the learning process finishes, it yields the WPLL $l_{F,s}$ of the acquired MLN with respect to the source data (line 6). We then rescale the WPLL to lie between 0 and 1 as different domains can have different ranges of WPLLs, and denote the obtained value as the usefulness score $s_{F,s}$ (line 7).

Intuitively, $s_{F,s}$ reflects how much including $F_{T,s}$ into an MLN helps the model's discriminative power. To estimate the usefulness score $s_{T,s}$ of a template T, we average the usefulness scores $s_{F,s}$ across $\mathscr{F}_S$. Thus, we define the approximate sampling probabilities for each template T as $\hat{p}_{T,s} \sim p_T^0 \cdot s_{T,s}$ (line 9).

In the target domain, we similarly compute a probability $\hat{p}_{F,t}$ for each formula $F_{T,t}$, which estimates that formula's probability of inclusion in the target domain model. In a first step, we compute a usefulness score $s_{F,t}$ for each formula $F_{T,t}$ using the same procedure as in the source domain (line 11). In a second step, crucially, we multiply the resulting usefulness score with $\hat{p}_{T,s}$, the posterior probability of inclusion of the corresponding second-order template learned from the source domain (line 13).

5.3.2 Target-Domain Learning

TODTLER builds the target domain model $M_t^{(1)^*}$ incrementally, starting from an empty one in the following way. It arranges the formulas in $\bigcup_{T \in \mathscr{T}} \mathscr{F}_T$ in order of decreasing approximate probability $\hat{p}_{F,t}$ (line 18). For each formula in the ordered list, the approximation procedure attempts to add that formula to $M_t^{(1)^*}$, jointly learning the weights of all already included formulas and computing the model's WPLL (lines 20-21). A formula is added to the model only if it increases the WPLL of $M_t^{(1)^*}$ with respect to the target data (lines 22-25).

5.4 Experimental Evaluation

Our experiments compare the performance of TODTLER to DTM, which is the state-of-the-art transfer learning approach for relational domains (Davis and Domingos 2009). We also compare to learning from scratch using LSM, which is the state-of-the-art MLN structure learning algorithm (Kok and Domingos 2010). We evaluate the performance of the algorithms using data from three domains and address the following four research questions:

- Does TODTLER learn more accurate models than DTM?
- Does TODTLER learn more accurate models than LSM?
- Is TODTLER faster than DTM?

- Does TODTLER discover interesting SOLT templates?

5.4.1 Datasets

We use three datasets of which the first two have been widely used and are publicly available.[1] The **Yeast** protein dataset comes from the MIPS[2] Comprehensive Yeast Genome Database (Davis, Burnside, et al. 2005; Mewes et al. 2000). The dataset includes information about protein location, function, phenotype, class, and enzymes. The **WebKB** dataset consists of labeled web pages from the computer science departments of four universities (Craven and Slattery 2001). The dataset includes information about links between web pages, words that appear on the web pages, and the classifications of the pages. The **Twitter**[3] dataset contains tweets about Belgian soccer matches. The dataset includes information about follower relations between accounts, words that are tweeted, and the types of the accounts.

Table 5.1 reports the number of types, predicates, constants, true ground atoms, and possible ground atoms in all three domains.

	Twitter	WebKB	Yeast
Types	3	3	7
Predicates	3	3	7
Constants	378	4,396	3,105
True ground atoms	3,142	50,432	15,015
Possible ground atoms	53,748	4,732,804	1,387,014

Table 5.1: An overview of the characteristics of the datasets showing, for each dataset, the number of types, predicates, constants, true ground atoms, and possible ground atoms.

5.4.2 Methodology

Each of the datasets is a graph, which is divided into databases consisting of connected sets of facts (Mihalkova, Huynh, et al. 2007). Yeast and WebKB consist of four databases while Twitter consists of two. We trained each learner on a

[1] The Yeast and WebKB datasets are available on http://alchemy.cs.washington.edu.
[2] Munich Information Center for Protein Sequence
[3] The dataset is available on http://dtai.cs.kuleuven.be/software/todtler.

subset of the databases and tested it on the remaining databases. We repeated this cycle for all subsets of the databases.

We transferred with both TODTLER and DTM in all six possible source-target settings. Within each domain, both transfer learning algorithms had the same parameter settings. In each domain, we optimized the WPLL for the predicate of interest. We learned the weights of the formulas in the target model using Alchemy (Kok, Sumner, et al. 2010) and applied a pruning threshold of 0.05 on the weights of the clauses.

For DTM, we generated all clauses containing at most three literals and three object variables, and transferred five and ten second-order cliques to the target domain. Since DTM's refinement step can be computationally expensive, we limited its runtime to 100 hours per database.

For TODTLER, we enumerated all second-order templates containing at most three literals and three object variables. We assumed a uniform prior distribution over the second-order templates in the source domain, which means TODTLER's p_T^0 parameter was set to 0.5 for each template.

To evaluate each system, we jointly predict the truth value of all groundings of the Function predicate in Yeast, the PageClass predicate in WebKB, and the AccountType predicate in Twitter given evidence about all other predicates. We computed the probabilities using MC-SAT. After a burn-in of 1,000 samples, we computed the probabilities with the next 10,000 samples.[4]

We measured the area under the precision-recall curve (AUCPR) and the test set conditional log-likelihood (CLL) for the predicate of interest. CLL measures the quality of the probability estimates. AUCPR gives an indication of the predictive accuracy of the learned model. Furthermore, AUCPR is insensitive to the large number of true negatives in these domains. Unlike in ROC space, random guessing in PR space does not always correspond to a value of 0.50 but is skew-dependent (Boyd et al. 2012). In our experimental setup, random guessing yields an AUCPR of 0.07 for WebKB, 0.08 for Yeast, and 0.37 for Twitter.

We report the average relative difference in terms of AUCPR and CLL between the different methods. We compute these average relative differences as follows:

[4]The DTM paper performs leave-one-grounding-out inference while this chapter jointly infers all groundings of the target predicate.

$$\text{Relative difference AUCPR} = \frac{\text{AUCPR}_{\text{TODTLER}} - \text{AUCPR}_{\text{DTM}}}{\text{AUCPR}_{\text{TODTLER}}}$$

$$\text{Relative difference CLL} = \frac{\text{CLL}_{\text{DTM}} - \text{CLL}_{\text{TODTLER}}}{\text{CLL}_{\text{TODTLER}}}$$

5.4.3 Results

This section presents a comparison of TODTLER to the baseline methods in terms of accuracy, learning curves for all methods, a run time analysis, and the top-ranked templates discovered by TODTLER in each domain.

Comparison of TODTLER to the Baseline Methods

Tables 5.2 and 5.3 present the average relative differences in AUCPR between TODTLER and transferring five (DTM-5) as well as ten (DTM-10) cliques with DTM. The tables show the average relative differences for increasing amounts of training data. For example, the "3 DB" column presents the results for training on all subsets of three databases and evaluating on the remaining database. The N/A entries arise because the Twitter dataset only contains two databases. Positive average relative differences denote settings where TODTLER outperforms DTM. In terms of AUCPR, TODTLER outperforms both DTM-10 and DTM-5 in all 14 settings. In terms of CLL, TODTLER outperforms DTM-10 in 12 of the 14 settings and DTM-5 in 11 of the 14 settings.

Tables 5.4 and 5.5 present the average relative differences in AUCPR and CLL between TODTLER and LSM as well as DTM-5 and LSM. These experiments represent a comparison between performing transfer as opposed to learning from scratch in each domain with the state-of-the-art structure learner LSM.[5] TODTLER outperforms LSM in all 14 settings in terms of both AUCPR and CLL. DTM-5 outperforms LSM in 12 of the 14 settings in terms of both AUCPR and CLL. Hence, transferring knowledge from a source task leads to more accurate learned models than simply learning from scratch in the target domain.

[5]We picked DTM-5 as it generally exhibits better performance than DTM-10.

	AUCPR			CLL		
	1 DB	**2 DB**	**3 DB**	**1 DB**	**2 DB**	**3 DB**
WebKB → Yeast	0.231	0.344	0.540	-0.097	1.137	2.112
Twitter → Yeast	0.171	0.353	0.260	-0.065	0.748	-0.127
Yeast → WebKB	0.479	0.607	0.627	0.121	0.095	0.191
Twitter → WebKB	0.578	0.587	0.605	1.055	0.155	0.156
WebKB → Twitter	0.150	N/A	N/A	5.140	N/A	N/A
Yeast → Twitter	0.152	N/A	N/A	5.469	N/A	N/A

Table 5.2: The average relative differences in AUCPR and CLL between TODTLER and DTM-5 as function of the amount of training data. Positive differences indicate settings where TODTLER outperforms DTM-5. TODTLER outperforms DTM-5 in all 14 settings in terms of AUCPR and in 11 of the 14 settings in terms of CLL. The N/A entries arise because the Twitter dataset only contains two databases.

	AUCPR			CLL		
	1 DB	**2 DB**	**3 DB**	**1 DB**	**2 DB**	**3 DB**
WebKB → Yeast	0.213	0.390	0.331	-0.114	0.650	0.517
Twitter → Yeast	0.190	0.325	0.362	-0.063	0.126	0.017
Yeast → WebKB	0.479	0.614	0.638	0.121	0.098	0.196
Twitter → WebKB	0.578	0.596	0.607	1.055	0.158	0.157
WebKB → Twitter	0.224	N/A	N/A	3.945	N/A	N/A
Yeast → Twitter	0.226	N/A	N/A	4.210	N/A	N/A

Table 5.3: The average relative differences in AUCPR and CLL between TODTLER and DTM-10 as function of the amount of training data. Positive differences indicate settings where TODTLER outperforms DTM-10. TODTLER outperforms DTM-10 in all 14 settings in terms of AUCPR and in 12 of the 14 settings in terms of CLL. The N/A entries arise because the Twitter dataset only contains two databases.

	AUCPR			CLL		
	1 DB	**2 DB**	**3 DB**	**1 DB**	**2 DB**	**3 DB**
WebKB → Yeast	0.471	0.671	0.583	5.075	12.156	8.841
Twitter → Yeast	0.479	0.676	0.589	6.091	14.356	10.486
Yeast → WebKB	0.576	0.561	0.562	0.079	0.073	0.070
Twitter → WebKB	0.576	0.561	0.562	0.072	0.066	0.064
WebKB → Twitter	0.599	N/A	N/A	13.463	N/A	N/A
Yeast → Twitter	0.600	N/A	N/A	14.238	N/A	N/A

Table 5.4: The average relative differences in AUCPR and CLL between TODTLER and LSM as function of the amount of training data. Positive differences indicate settings where TODTLER outperforms LSM. TODTLER outperforms LSM in all 14 settings in terms of both AUCPR and CLL. The N/A entries arise because the Twitter dataset only contains two databases.

	AUCPR			CLL		
	1 DB	**2 DB**	**3 DB**	**1 DB**	**2 DB**	**3 DB**
WebKB → Yeast	0.311	0.518	0.161	5.725	6.537	3.328
Twitter → Yeast	0.371	0.542	0.459	6.584	8.764	12.832
Yeast → WebKB	0.186	-0.018	0.032	-0.037	0.001	0.006
Twitter → WebKB	-0.004	0.003	0.058	-0.478	0.003	0.007
WebKB → Twitter	0.528	N/A	N/A	1.355	N/A	N/A
Yeast → Twitter	0.528	N/A	N/A	1.355	N/A	N/A

Table 5.5: The average relative differences in AUCPR and CLL between DTM-5 and LSM as function of the amount of training data. Positive differences indicate settings where DTM-5 outperforms LSM. DTM-5 outperforms LSM in 12 of the 14 settings in terms of both AUCPR and CLL. The N/A entries arise because the Twitter dataset only contains two databases.

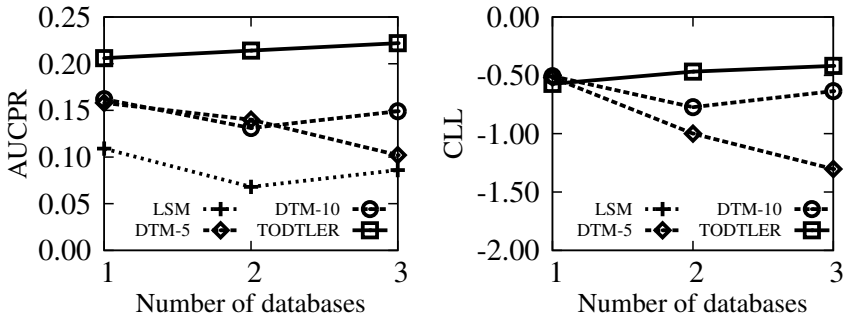
Learning Curves for All Methods

Figure 5.2 presents learning curves for predicting protein function in the Yeast dataset when transferring from WebKB. TODTLER outperforms DTM-10, DTM-5 and LSM in terms of both AUCPR and CLL. Since LSM obtains a much worse CLL than the other systems, its curve falls out of the range of the graph.

Figure 5.3 presents learning curves for predicting protein function in the Yeast dataset when transferring from Twitter. TODTLER outperforms DTM-10, DTM-5 and LSM in terms of AUCPR and exhibits similar performance as DTM-10 in terms of CLL. Since LSM obtains a much worse CLL than the other systems, its curve falls out of the range of the graph.

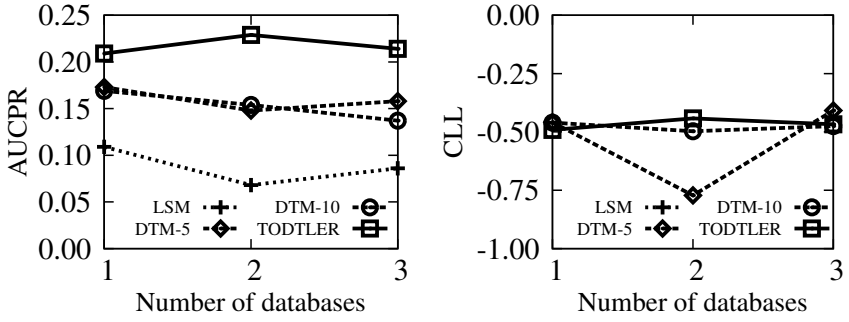
Figure 5.4 presents learning curves for predicting a web page's class in the WebKB dataset when transferring from Yeast. TODTLER outperforms DTM-10, DTM-5 and LSM in terms of both AUCPR and CLL. TODTLER's performance improves for an increasing amount of training data.

Figure 5.5 presents learning curves for predicting a web page's class in the WebKB dataset when transferring from Twitter. TODTLER outperforms DTM-10, DTM-5 and LSM in terms of both AUCPR and CLL. TODTLER's performance improves for an increasing amount of training data.



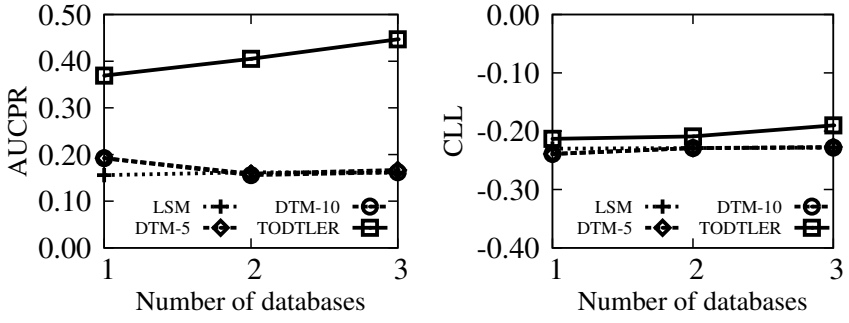
(a) AUCPR (b) CLL

Figure 5.2: The learning curves for predicting protein function in Yeast when transferring knowledge from WebKB.



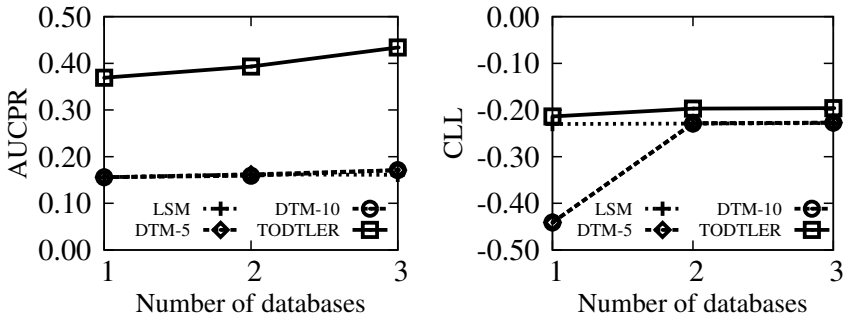
(a) AUCPR (b) CLL

Figure 5.3: The learning curves for predicting protein function in Yeast when transferring knowledge from Twitter.



(a) AUCPR (b) CLL

Figure 5.4: The learning curves for predicting a web page's class in WebKB when transferring knowledge from Yeast.



(a) AUCPR (b) CLL

Figure 5.5: The learning curves for predicting a web page's class in WebKB when transferring knowledge from Twitter.

Runtime Analysis

In addition to learning more accurate models, TODTLER also exhibits a much faster runtime than DTM as is shown in Tables 5.6 and 5.7. Across all the considered settings, TODTLER is 8 to 44 times faster in Yeast, 5 to 29 times faster in WebKB, and 132 to 264 times faster in Twitter.

A couple of reasons contribute to TODTLER's improved runtime. First, for learning in the target domain, DTM runs an iterative greedy strategy that picks the single best candidate formula in each step. This is more expensive than TODTLER's non-iterative target-domain strategy for picking formulas. Second, DTM performs a refinement step, which improves accuracy but is computationally costly as it is another greedy search approach.

Templates Discovered by TODTLER

Tables 5.8, 5.9, and 5.10 present the ten top-ranked second-order templates in the Yeast, WebKB, and Twitter domains, respectively.

One example is $R_1(X, Y) \vee \neg R_1(Y, X)$, which represents symmetry and ranks first in Yeast and WebKB and second in Twitter. When transferred to the Twitter problem, this template gives, among others, rise to the first-order

	TODTLER			DTM-5		
	1 DB	**2 DB**	**3 DB**	**1 DB**	**2 DB**	**3 DB**
WebKB → Yeast	103	199	338	1,766	6,234	14,896
Twitter → Yeast	93	181	277	707	2,496	9,555
Yeast → WebKB	16	26	45	84	478	753
Twitter → WebKB	13	21	44	122	294	464
WebKB → Twitter	1	N/A	N/A	49	N/A	N/A
Yeast → Twitter	1	N/A	N/A	50	N/A	N/A

Table 5.6: The average runtimes in minutes for TODTLER and DTM-5. TODTLER is consistently faster than DTM-5. The N/A entries arise because the Twitter dataset only contains two databases.

	TODTLER			DTM-10		
	1 DB	**2 DB**	**3 DB**	**1 DB**	**2 DB**	**3 DB**
WebKB → Yeast	103	199	338	1,759	6,671	9,206
Twitter → Yeast	93	181	277	725	1,683	4,635
Yeast → WebKB	16	26	45	95	571	1,323
Twitter → WebKB	13	21	44	142	392	840
WebKB → Twitter	1	N/A	N/A	75	N/A	N/A
Yeast → Twitter	1	N/A	N/A	76	N/A	N/A

Table 5.7: The average runtimes in minutes for TODTLER and DTM-10. TODTLER is consistently faster than DTM-10. The N/A entries arise because the Twitter dataset only contains two databases.

formula $\text{Follows}(X, Y) \vee \neg\text{Follows}(Y, X)$, meaning that if an account Y follows an account X, X is likely to follow Y as well.

Another example is $R_1(X, Y) \vee \neg R_1(Z, Y) \vee \neg R_2(X, Z)$, which ranks third in Yeast, eighth in WebKB and ninth in Twitter. This template represents the concept of homophily, which means that related objects (X and Z) tend to have similar properties (Y). This template gives, among others, rise to the first-order formula $\text{Has}(X, Y) \vee \neg\text{Has}(Z, Y) \vee \neg\text{Linked}(X, Z)$ when transferred to the WebKB problem. This formula means that if a web page X links to a web page Z, both web pages are likely to contain the same word Y.

#	Template
1	$R_1(X,Y) \vee \neg R_1(Y,X)$
2	$R_1(X,Y) \vee \neg R_1(Y,X) \vee \neg R_2(X,Z)$
3	$R_1(X,Y) \vee \neg R_1(Z,Y) \vee \neg R_2(X,Z)$
4	$R_1(X,Y) \vee \neg R_1(X,Z) \vee \neg R_1(Y,X)$
5	$\neg R_1(X,Y) \vee R_1(Y,X) \vee \neg R_2(X,Z)$
6	$\neg R_1(X,Y) \vee R_1(X,Z) \vee \neg R_1(Y,Z)$
7	$\neg R_1(X,Y) \vee R_1(Z,Y) \vee \neg R_2(X,Z)$
8	$R_1(X,Y) \vee \neg R_1(Y,X) \vee \neg R_1(Z,X)$
9	$R_1(X,Y) \vee \neg R_1(Y,Z) \vee \neg R_1(Z,X)$
10	$R_1(X,Y) \vee \neg R_1(X,Z) \vee \neg R_1(Y,Z)$

Table 5.8: The ten top-ranked second-order templates in the Yeast domain.

#	Template
1	$R_1(X,Y) \vee \neg R_1(Y,X)$
2	$\neg R_1(X,Y) \vee R_1(X,Z) \vee \neg R_1(Y,Z)$
3	$\neg R_1(X,Y) \vee \neg R_1(Y,X) \vee R_2(X,Z)$
4	$\neg R_1(X,Y) \vee R_1(Y,Z) \vee \neg R_1(Z,X)$
5	$\neg R_1(X,Y) \vee \neg R_1(X,Z) \vee R_1(Y,Z)$
6	$R_1(X,Y) \vee \neg R_1(X,Z) \vee \neg R_1(Y,X)$
7	$\neg R_1(X,Y) \vee \neg R_1(Z,Y) \vee R_2(X,Z)$
8	$R_1(X,Y) \vee \neg R_1(Z,Y) \vee \neg R_2(X,Z)$
9	$\neg R_1(X,Y) \vee \neg R_1(X,Z) \vee R_1(Y,X)$
10	$R_1(X,Y) \vee \neg R_1(Y,X) \vee \neg R_1(Z,X)$

Table 5.9: The ten top-ranked second-order templates in the WebKB domain.

#	Template
1	$R_1(X,Y) \vee \neg R_1(X,Z)$
2	$R_1(X,Y) \vee \neg R_1(Y,X)$
3	$\neg R_1(X,Y) \vee \neg R_1(Y,X) \vee R_1(Z,X)$
4	$\neg R_1(X,Y) \vee R_1(X,Z) \vee \neg R_1(Y,X)$
5	$R_1(X,Y) \vee \neg R_1(X,Z) \vee \neg R_1(Y,Z)$
6	$\neg R_1(X,Y) \vee R_1(Z,Y) \vee \neg R_2(X,Z)$
7	$\neg R_1(X,Y) \vee \neg R_1(Y,X) \vee R_2(X,Z)$
8	$\neg R_1(X,Y) \vee R_1(Y,X) \vee \neg R_1(Z,X)$
9	$R_1(X,Y) \vee \neg R_1(Z,Y) \vee \neg R_2(X,Z)$
10	$R_1(X,Y) \vee \neg R_1(X,Z) \vee \neg R_1(Y,X)$

Table 5.10: The ten top-ranked second-order templates in the Twitter domain.

Discussion

Several possible explanations exist about why TODTLER learns more accurate models than DTM. First, TODTLER transfers fine-grained knowledge because it performs transfer on a per-template basis instead of a per-clique basis. As discussed in the background, DTM's second-order cliques group together multiple second-order formulas. Then, each of these second-order formulas gives rise to one or multiple first-order formulas. Within a clique, only a small subset of these formulas will be helpful for modeling the target domain. Second, TODTLER transfers both the second-order templates (i.e., structural regularities) as well as information about their usefulness (i.e., the posterior of the formulas, $P_s(M^{(2)}|D_s)$). In contrast, DTM just transfers the second-order cliques and the target data is used to assess whether the regularities are important. Finally, TODTLER transfers a more diversified set of regularities whereas DTM is restricted to a smaller set of user-selected cliques. This increases the chance that TODTLER transfers something of use for modeling the target domain.

5.5 Conclusions

This chapter investigated the task of performing transfer learning in relational domains and presented a novel framework called TODTLER that addresses this task. More specifically, this chapter focused on the deep-transfer-learning task, where the source and target domains can consist of entirely different sets of entities and relationships, in the context of Markov logic networks.

The proposed TODTLER framework views knowledge transfer as the process of learning a declarative bias in one domain and transferring it to another to improve the learning process. TODTLER applies a two-stage procedure, which discovers patterns that are useful in the source domain and biases the learning process in the target domain towards models that have these patterns as well.

Our extensive empirical evaluation on three real-world datasets shows that TODTLER outperforms the previous state-of-the-art deep-transfer-learning approach DTM as well as the state-of-the-art inductive-learning algorithm LSM in terms of accuracy. While producing more accurate models, TODTLER is also significantly faster than DTM and LSM.

In the future, it may be possible to improve TODTLER's performance further

by enabling it to transfer a richer set of patterns and regularities than any deep-transfer-learning algorithm can currently handle. Furthermore, it would be worth investigating whether TODTLER could be used as a stand-alone structure-learning algorithm. Moore and Danyluk (2010) presented some evidence that DTM is well-suited for that task.

6

Discovering Offensive Strategies in Soccer Data

Moneyball (Lewis 2004) tells the story of Oakland A's General Manager Billy Beane who relies on statistics to build a competitive baseball team. Despite a tight budget, Beane achieved to assemble a successful team that manages to win a record-number of 20 consecutive matches. In recent years, Beane's work has been an example for many other ball sports like basketball, football, and soccer. While several aspects of baseball matches can be analyzed in a rather straightforward way, analyzing more continuous sports where players can freely move around the pitch is much harder. As a result, objectively quantifying the performances of individual players and teams can be very challenging.

Although simple statistics such as the number of shots on target in soccer can easily be collected, they fail to capture the complex movements and interactions among the players on the pitch. Therefore, companies such as ChyronHego,[1] Prozone,[2] and STATS[3] have developed optical tracking systems that capture the locations of the players and the ball at a high frequency. These positional data do not only tell how often a particular event happened in a match but also when, where, and how the event happened. While many professional sports clubs

[1] ChyronHego TRACAB: http://www.chyronhego.com/sports-data/tracab
[2] Prozone: http://prozonesports.stats.com
[3] STATS SportVU: http://www.stats.com/sportvu-basketball/

have access to large volumes of performance data, the valuable information that is hidden in these data is only used to a limited extent. Sports clubs simply lack the computational methods to analyze these data in greater depth.

This chapter proposes and addresses the task of automatically discovering offensive strategies in professional soccer matches. More specifically, the objective is to automatically reveal the player interactions that are most likely to lead to a goal attempt. This is a challenging task due to the low-scoring and continuous nature of soccer matches. This chapter introduces an inductive-logic-programming approach that can deal with the relational data that is omnipresent in sports domains in a natural way.

Contributions of this Chapter

The first contribution of this chapter is an inductive-logic-programming approach that automatically discovers offensive strategies in spatio-temporal soccer match data. More specifically, the approach aims to discover patterns that occur more often than not in phases leading to a goal attempt.

The second contribution is an empirical study on a large volume of soccer matches that was acquired through a collaboration with a Belgian professional soccer club. The study demonstrates that the proposed approach is able to automatically discover interesting and relevant offensive strategies.

The content of this chapter is based on the following publication:

Jan Van Haaren, Vladimir Dzyuba, Siebe Hannosset, and Jesse Davis (2015). "Automatically Discovering Offensive Patterns in Soccer Match Data". In: *Advances in Intelligent Data Analysis XIV (IDA 2015; Saint-Étienne, France; 22-24 October 2015)*, pages 286–297

The methodology and experimental study are the work of the author of this dissertation. The analysis of the experimental results is joint work between Vladimir Dzyuba and the author of this dissertation.

Structure of this Chapter

Section 6.1 provides relevant background on knowledge discovery and the analysis of sports data. **Section 6.2** describes the structure and content of the

dataset. **Section 6.3** introduces the proposed inductive-logic-programming approach. **Section 6.4** presents an empirical study on a large volume of soccer matches. **Section 6.5** lists several important lessons learned. **Section 6.6** provides conclusions and directions for future work.

6.1 Related Work

This section provides an overview of the related work on supervised knowledge discovery and sports analytics. The relevant background on inductive logic programming, which is the core of our approach, is provided in Section 2.1.2.

6.1.1 Knowledge Discovery

The problem addressed in this chapter is an instance of supervised descriptive rule discovery (Kralj Novak et al. 2009). A common variant of this problem is subgroup discovery (Herrera et al. 2011). Although early variants already supported multi-relational data (Wrobel 1997), the data are typically merged into a single table before applying subgroup discovery algorithms (Lavrač, Cestnik, et al. 2004). By contrast, inductive logic programming techniques allow us to work directly with the relational (logical) representation of data. This is important for our task, where we want to capture both spatial and temporal patterns as well as interactions among groups of players. An alternative perspective on relational data mining relies on database theory (Knobbe 2004).

6.1.2 Sports Data Analysis

The amount of available sports data is constantly increasing, most importantly tracking data and event data (Mutschler et al. 2013). Within soccer, the analysis of tracking data focuses on discovering individual or collective movement patterns, e.g., spectral clustering of trajectories (Knauf and Brefeld 2014), strategy analysis with occupancy maps (Lucey et al. 2013), or formation analysis via minimum entropy partitioning (Bialkowski et al. 2014). Gyarmati et al. (2014) use event data to discover motif patterns in pass sequences. Most of the research studies large datasets encompassing multiple teams or even leagues, whereas we focus on a single team, with the ultimate goal to improve its performance.

6.2 Dataset

Through our collaboration with a Belgian soccer club, we obtained play-by-play data for 70 soccer matches in the 2013/2014 and 2014/2015 seasons. The dataset consists of 59 matches in the Belgian Pro League, nine matches in the UEFA Europa League and two matches in the Belgian Cofidis Cup. The data were collected by data provider Prozone. We first discuss the structure of the data and then introduce additional hierarchical information to enrich the dataset.

6.2.1 Structure of the Data

The data for each match is provided as an XML file which consists of three parts: a match sheet with information on the players and managers, a sequence of events, and tracking data for all players as well as the ball. While the first two parts are available for all matches, the third part is only available for 10 Jupiler Pro League and 4 UEFA Europa League matches.

The match sheet contains each player's name, position on the pitch, jersey number, and team. In addition, it also specifies which players were starters and which players were substitutes.

The sequence of events contains roughly 2,600 events per match. Over 40 different types of events are recorded. The most frequent events include passes between players, players running with the ball, players receiving a ball, players shooting towards goal, players fouling another player, players crossing the ball, and players clearing the ball. Furthermore, events exist to mark the start and end of each half as well as yellow cards, red cards, and substitutions.

For each event, the following information is available: the type of the event, the players that are involved, a timestamp, the start location of the event, and the end location of the event if applicable. Depending on the type of event, additional information is available such as the body part involved (e.g., foot or head), type of play (i.e., open or set play), or whether or not a shot was blocked.

6.2.2 Hierarchical Information

Since we prefer more general patterns to very specific patterns, we enrich the dataset with hierarchical information about both the pitch and the players. This

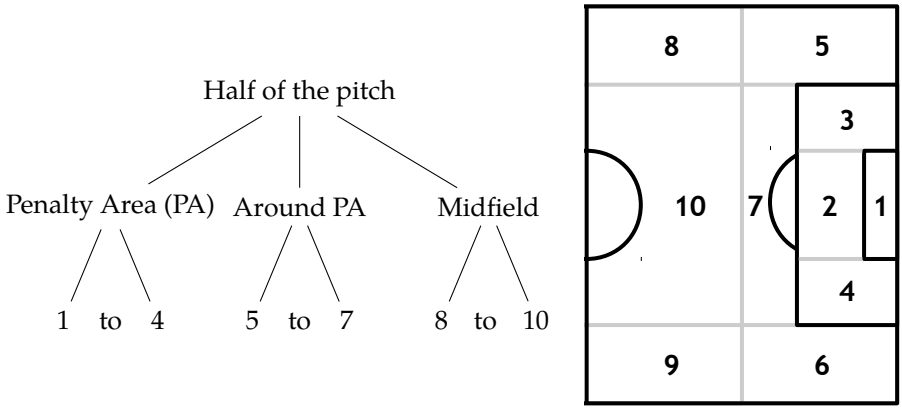


Figure 6.1: The division of the pitch into zones. Each half of the pitch is divided into ten zones, which we group together into three bigger areas. Zones 1 to 4 are the *penalty area*, zones 5 to 7 the *area around the penalty area*, and zones 8 to 10 the *midfield*. The division is identical for the defensive and offensive half.

information groups together parts of the pitch and players that fulfill a similar role and hence can be treated in a similar way. As a result, this information facilitates generalizing from very specific to more general knowledge.

We divide each half of the pitch into ten zones resulting into twenty different zones as is shown on the right side of Figure 6.1. Assuming the team of interest always plays from left to right, we define a hierarchy as follows. We group together zones 1 to 4 as the *penalty area*, zones 5 to 7 as the *area around the penalty area*, and zones 8 to 10 as the *midfield*. The division is identical for the defensive and offensive half of the pitch.

Similarly, we group together players that play in a similar position. We define four groups of players for the team of interest: *goalkeepers*, *defenders* (i.e., center backs, full backs, wing backs, and sweepers), *midfielders* (i.e., defensive midfielders, central midfielders, attacking midfielders, and wing midfielders), and *attackers* (i.e., wingers, supporting strikers, and strikers).

6.3 Methodology

This section introduces our inductive-logic-programming approach to automatically discover patterns that frequently appear in successful offensive strategies. We explain how we pre-process the data and learn the clauses.

6.3.1 Pre-processing the Data

As explained in Section 6.2.1, the dataset consists of one long sequence of events for each match. We split each sequence into a number of phases, each of which is a subsequence of related events. A phase typically starts with a goal kick or a throw-in and ends when the ball goes out of play or a foul is made. We only consider passes, crosses, set pieces and shots, and discard all other events. We also only consider phases in which the team of interest is dominant, which is when its players are involved in at least half of the events. Although this rarely happens, both teams can be seen as the dominant team in the same phase. However, this is not a problem since we are only looking at the team of interest.

Building Examples

In our setting, we define positive examples as phases during which the team of interest attempts a shot, and we label all other phases as a *negative* examples. Thus, the target predicate is $\texttt{shot}(\texttt{Phase})$, which denotes whether the team attempted a shot in a phase `Phase`. In the background knowledge, we represent each phase as a set of ground facts using four predicates. The $\texttt{pass}(\texttt{Phase},\texttt{Player}_1,\texttt{Player}_2,\texttt{Zone}_1,\texttt{Zone}_2)$ predicate denotes that in a phase `Phase` a player $\texttt{Player}_1$ in zone $\texttt{Zone}_1$ passed the ball to $\texttt{Player}_2$ in zone $\texttt{Zone}_2$. Similarly, the $\texttt{cross}(\texttt{Phase},\texttt{Player}_1,\texttt{Player}_2,\texttt{Zone}_1,\texttt{Zone}_2)$ and $\texttt{set\_piece}(\texttt{Phase},\texttt{Player}_1,\texttt{Player}_2,\texttt{Zone}_1,\texttt{Zone}_2)$ predicates denote crosses and set pieces. For positive examples, we discard all events following a shot.

Adding Background Knowledge

We add the hierarchical information about both the pitch and the players as background knowledge. The following are two examples of background knowledge for the `pass` predicate.

$$\texttt{pass}(\texttt{Ph}, \texttt{pl}_1, \texttt{pl}_2, \texttt{Z}_1, \texttt{Z}_2) \rightarrow \texttt{pass}(\texttt{Ph}, \texttt{pMidfielder}, \texttt{pAttacker}, \texttt{Z}_1, \texttt{Z}_2) \quad (6.1)$$

$$\texttt{pass}(\texttt{Ph}, \texttt{P}_1, \texttt{P}_2, \texttt{z}_2, \texttt{z}_7) \rightarrow \texttt{pass}(\texttt{Ph}, \texttt{P}_1, \texttt{P}_2, \texttt{zPenaltyArea}, \texttt{zMidfield}) \quad (6.2)$$

Assuming player $\texttt{pl}_1$ is a midfielder and player $\texttt{pl}_2$ is an attacker, Equation 6.1 denotes that if $\texttt{pl}_1$ passes the ball to $\texttt{pl}_2$, then also a midfielder passes the ball to an attacker. Assuming zone $\texttt{z}_2$ belongs to the penalty area and zone $\texttt{z}_7$ belongs to the midfield (see Figure 6.1), Equation 6.2 denotes that a player who passes the ball from $\texttt{z}_2$ to $\texttt{z}_7$ also passes the ball from the penalty area to midfield.

As a practical optimization akin to view materialization in databases, we specify the background knowledge like this rather than by adding additional predicates.

6.3.2 Learning the Clauses

The Aleph system supports many different learning modes and search strategies (Srinivasan 2001). We apply the `induce_max` search strategy. Unlike the default search strategy, this strategy uses each positive example as a seed example. While slower, it produces a larger set of clauses that are of potential interest to the user. However, this is a natural choice when doing exploratory data mining as our goal is to generate interesting clauses as opposed to learning a very compact predictive model, which is the traditional goal of ILP.

Since we are interested in as many potentially interesting clauses as possible, we run Aleph with as few restrictions as reasonably possible. We set the maximum number of literals per clause (i.e., `clauselength`) to 5, the minimum number of positive examples covered (i.e., `minpos`) to 5, the maximum number of negative examples covered (i.e., `noise`) to 25, and the minimum precision of acceptable clauses (i.e., `minacc`), which is the ratio between the number of positive examples covered and the total number of examples covered, to 5%.

We sort the learned clauses in descending order according to their m-estimates (Cestnik 1990; Lavrač, Džeroski, et al. 1996). The m-estimate of a rule can be viewed as a smoothed version of its precision.

6.4 Experimental Evaluation

In this section, we present the dataset as well as the different experimental setups, define the research questions, and discuss the experimental results.

6.4.1 Dataset and Experimental Setups

After pre-processing the raw data as described earlier, the dataset contains 3,803 examples (phases), including 526 (13.8%) positive examples (shots), and 26,338 ground facts in total, including 24,786 passes (94.1%), 1,063 crosses (4.0%), and 489 set pieces (1.9%). An average example consists of 6.93 ground facts, including 6.52 passes, 0.28 crosses, and 0.13 set pieces. Furthermore, there are 34 constants corresponding to the players of the team of interest.

We investigate the performance of the proposed approach in five setups: discovering spatial patterns with and without hierarchical information, player-interaction patterns with and without hierarchical information, and the combined setup with the hierarchical information, in order to evaluate the utility of each type of background knowledge.

6.4.2 Research Questions

This experimental study addresses the following three research questions:

- **Q1: Do the learned clauses capture the relevant regularities?** The ultimate goal of the analysis is to describe the team's successful offensive actions. We quantify the ability of the proposed approach to accomplish this by computing the average m-estimate of the top-ten clauses.
- **Q2: Does the hierarchical background knowledge improve the quality of the learned clauses?** One motivation for using ILP is its ability to represent relational data such as the player and zone hierarchies in a natural way. We investigate whether the addition of the hierarchies improves the quality of the learned clauses.
- **Q3: Do the learned clauses describe meaningful patterns?** The purpose of this work is to discover patterns that help the team understand what

works well and what does not work well in terms of creating goal-scoring opportunities. Therefore, we qualitatively analyze the discovered patterns.

The proposed approach is meant to facilitate offline performance analysis, e.g., between matches or even seasons. Therefore, it is not necessary to produce instant results. All experiments are run on a single core of a Linux machine with an Intel Xeon E5645 CPU running at 2.40 GHz and 128 Gb of RAM. We allow the Aleph system to run for 48 hours in each setup.

6.4.3 Experiments

We first address research questions Q1 and Q2 by comparing the five different setups using statistics on the sets of discovered clauses. We then address research question Q3 by evaluating the utility of the clauses for the first four setups from a performance analysis point of view.

6.4.4 Quantitative Analysis (Q1 and Q2)

Table 6.1 contains an overview of the experimental results. We expect that adding hierarchical background knowledge allows Aleph to find clauses of higher quality. We observe a considerable improvement in terms of average m-estimate in the player-interaction setup, while this increase is rather modest in the spatial setup. However, the runtime cost of adding hierarchical information is substantial since the search space becomes much larger. In the player-interaction setup, Aleph still manages to explore the whole search space and to generate high-quality candidate clauses in terms of m-estimate, which it fails to accomplish in the combined setup.

6.4.5 Qualitative Analysis (Q3)

Tables 6.2 and 6.3 present the top-three clauses in terms of their m-estimates for discovering spatial patterns both with and without hierarchical information. These settings have two of their three top-ranked clauses in common (i.e., clauses A and B). Clause A describes a situation where the ball is passed between two players in the left defensive zone (`d5`), from the defensive midfield (`d10`) to the right offensive wing (`o9`), and between two players in

Setup	Hierarchy	Rules	Max. m-est.	Avg. m-est.	Time
Spatial	✗	276	0.7396	0.6638	1.15
Spatial	✓	323	0.7396	0.7065	441.76
Player interactions	✗	91	0.7396	0.4855	2.95
Player interactions	✓	257	0.7396	0.6606	2,761.64
Combined	✓	(⋆) 426	0.6374	0.6138	2,880.00

Table 6.1: An overview of the number of rules, the maximum and average m-estimate of the precision for the top-ten rules, and the runtime in minutes for each setup. In the setup marked by (⋆), Aleph exceeds the runtime threshold of 48 hours. Hence, the m-estimate is computed on the intermediate output.

	Clause $(\mathbf{C})$	$\lvert\mathbf{C}\rvert$	$\lvert\mathbf{C}^+\rvert$
A	$\texttt{pass}(\texttt{d10},\texttt{o9}) \wedge \texttt{pass}(\texttt{d5},\texttt{d5}) \wedge \texttt{pass}(\texttt{o10},\texttt{o10})$	5	5
B	$\texttt{pass}(\texttt{d10},\texttt{d8}) \wedge \texttt{pass}(\texttt{d10},\texttt{o8}) \wedge \texttt{pass}(\texttt{d6},\texttt{d6})$	5	5
C	$\texttt{pass}(\texttt{d10},\texttt{o9}) \wedge \texttt{pass}(\texttt{d5},\texttt{d8}) \wedge \texttt{pass}(\texttt{o10},\texttt{o7}) \wedge \texttt{pass}(\texttt{o9},\texttt{o10})$	5	5

Table 6.2: An overview of the top-three rules in terms of their m-estimates for discovering spatial patterns without hierarchical background information. For each rule, we report the total number of examples covered and the number of positive examples covered.

	Clause $(\mathbf{C})$	$\lvert\mathbf{C}\rvert$	$\lvert\mathbf{C}^+\rvert$
D	$\texttt{pass}(\texttt{d6},\texttt{d9}) \wedge \texttt{pass}(\texttt{dAPA},\texttt{dPA}) \wedge \texttt{pass}(\texttt{o10},\texttt{o7})$	5	5
A	$\texttt{pass}(\texttt{d10},\texttt{o9}) \wedge \texttt{pass}(\texttt{d5},\texttt{d5}) \wedge \texttt{pass}(\texttt{o10},\texttt{o10})$	5	5
B	$\texttt{pass}(\texttt{d10},\texttt{d8}) \wedge \texttt{pass}(\texttt{d10},\texttt{o8}) \wedge \texttt{pass}(\texttt{d6},\texttt{d6})$	5	5

Table 6.3: An overview of the top-three rules in terms of their m-estimates for discovering spatial patterns with hierarchical background information. For each rule, we report the total number of examples covered and the number of positive examples covered.

	Clause ($\mathbf{C}$)	$\lvert\mathbf{C}\rvert$	$\lvert\mathbf{C}^+\rvert$
A	pass(p1,p21) ∧ pass(p8,p18)	5	5
B	pass(p18,p9) ∧ pass(p2,p18)	6	5
C	pass(p2,p26) ∧ pass(p3,p1)	8	5

Table 6.4: An overview of the top-three rules in terms of their m-estimates for discovering player-interaction patterns without hierarchical background information. For each rule, we report the total number of examples covered and the number of positive examples covered.

	Clause ($\mathbf{C}$)	$\lvert\mathbf{C}\rvert$	$\lvert\mathbf{C}^+\rvert$
A	pass(p1,p21) ∧ pass(p8,p18)	5	5
D	pass(att,att) ∧ pass(mid,att) ∧ pass(mid,def) ∧ pass(p4,p16)	5	5
E	pass(def,att) ∧ pass(def,mid) ∧ pass(opp,p2) ∧ pass(p8,opp)	7	6

Table 6.5: An overview of the top-three rules in terms of their m-estimates for discovering player-interaction patterns with hierarchical background information. For each rule, we report the total number of examples covered and the number of positive examples covered.

the offensive midfield (o10). Clause B describes a situation where the ball is passed between two players in the right defensive zone (d6) and from the defensive midfield (d10) to both the left defensive wing (d8) and the left offensive wing (o8). Both clauses suggest that the team is particularly successful at creating goal attempts when moving the ball from one flank to the other.

Clause D, which leverages the hierarchical information, describes a situation where the ball is passed from the area around the defensive penalty area (dAPA) into the defensive penalty area (dPA), from the right defensive zone (d6) to the right defensive wing (d9), and from the offensive midfield (o10) to the central offensive area around the penalty area (o7). This pattern most probably depicts a counter-attack following a set piece from the opponent.

Tables 6.4 and 6.5 present the top-three clauses in terms of their m-estimates for discovering player-interaction patterns both with and without hierarchical information. These settings have only one of their three top-ranked clauses in common (i.e., clause A). Clause A describes a situation where the goalkeeper (p1) passes the ball to a central defender (p21) and an attacking midfielder (p8) passes

	Clause (**C**)	$\|\mathbf{C}\|$	$\|\mathbf{C}^+\|$	WRA	m-est.
A	pass(oMF, oMF) ∧ pass(oMF, oPA) ∧ pass(oPA, oAPA)	62	18	0.025	0.275
B	pass(o4, o7)	43	15	0.024	0.324
C	set_piece(dAPA, dPA)	51	16	0.024	0.295

Table 6.6: An overview of the top-three rules in terms of their weighted relative accuracies for discovering spatial patterns with hierarchical background information. For each rule, we report the weighted relative accuracy and m-estimate. These rules are more general and less pure than the top-ranked rules according to m-estimate for the same setup.

the ball to an offensive wing midfielder (p18). This pattern makes sense from a sports perspective as both p8 and p18 are generally considered key players and responsible for creating a large number of goal-scoring opportunities.

Clause B describes a situation where an offensive full back (p2) passes the ball to an offensive wing midfielder (p18) and the latter player passes the ball to another wing midfielder (p9). This pattern makes sense as well as p2 has had a foot in many goals scored by the team of interest. Clause C describes a similar pattern involving a goalkeeper (p1), a central defender (p3), an offensive full back (p2), and a central midfielder (p26).

Clauses D and E leverage the hierarchical information about player roles as they include both specific players (e.g., p4 and p16) and positions (e.g., mid and att). Clause D describes an attack over the left wing involving both an offensive full back (p4) and an offensive wing midfielder (p16), while clause E describes a situation where an offensive full back (p2) intercepts a pass from an opponent (opp) and an attacking midfielder (p8) attempts a possibly risky pass that is briefly intercepted or touched by an opponent.

6.4.6 Alternative Qualitative Analysis (Q3)

We observed that the top-ranked clauses according to m-estimate are markedly specific. Therefore, we compare these clauses with the top-ranked clauses in the same set of clauses according to *weighted relative accuracy*, which is a common quality measure that aims to balance rule coverage and specificity:

$$WRA(C) = \frac{|C|}{|E|} \cdot \left(\frac{|C^+|}{|C|} - \frac{|E^+|}{|E|} \right)$$

Table 6.6 presents the top-three clauses in terms of WRA for discovering spatial patterns with hierarchical knowledge. These patterns have a substantially higher coverage, while their m-estimates are much lower. In the same setup, the average m-estimate for the top-ten clauses was 0.707. This contrasts with Op De Beéck et al. (2015), where in a similar setting, the coverage of the top-ranked clauses according to m-estimate ranges from 30 to 90 examples. This suggests that different quality measures could reveal different patterns in a dataset. Therefore, if the initial results are unsatisfactory from the domain perspective, ranking the clauses with another quality measure is a reasonable next step.

Clause A describes an attack through the middle, where the ball is passed between two players in the offensive midfield (`oMF`), from the offensive midfield to the offensive penalty area (`oPA`), and from the offensive penalty area to the area around the offensive penalty area (`oAPA`). Clause B describes a pass from the right side of the offensive penalty area (`o4`) to the area in front of the offensive penalty area (`o7`). Clause C describes a set piece from the area around the defensive penalty area (`dAPA`) into the defensive penalty area (`dPA`). Hence, this clause describes a situation where a counter-attack results in a goal-scoring opportunity. These patterns are different from those in Tables 6.2 and 6.3.

6.5 Lessons Learned

While undertaking this study, we learned the following three lessons.

First, it is possible to apply inductive logic programming to the task of revealing recurring patterns in soccer match data. It provides the advantages of coping with the relational nature of the data in a straightforward way. Furthermore, it produces interpretable results, which facilitates debugging the data, analyzing the results, and communicating with domain experts.

Second, the discovered patterns make sense from a soccer perspective and are interesting to a domain expert. However, taking the next step forward would require the full tracking data (i.e., the positions of the players and the ball at

regular intervals) as this will allow for more fine-grained analysis. Fortunately, this type of data is becoming commonplace.

Third, selecting the most interesting clauses is difficult as there is no natural metric or heuristic for this task. A human domain expert is still needed to assist in the interpretation of the discovered patterns.

6.6 Conclusions

This chapter investigated the task of automatically discovering recurring patterns in successful offensive strategies in soccer matches and presented an inductive-logic-programming approach that addresses this task.

The proposed approach views a soccer match as a long stream of events and performs the following three steps. First, the approach splits each match into a set of phases, which are subsequences of related events. Second, it labels the phases that lead to a goal attempt as positive and all others as negative. Third, the approach learns rules describing patterns that appear more often in phases labeled positive than in phases labeled negative.

Our experimental evaluation on a large number of matches from a Belgian professional soccer club shows that the proposed approach is able to discover both spatial (e.g., a pass from one zone to another) and player-interaction (e.g., a pass from one player to another) patterns that are likely to lead to goal attempts.

In the future, it may be possible to improve the performance of the approach by explicitly taking into account the order of the events, the positions of the players as well as the ball, and the differences in playing style of the different opponents. Furthermore, it would be helpful to develop a tool that visualizes the discovered patterns (e.g., on a soccer pitch as partially shown in Figure 6.1) to help communicate the discovered patterns to a domain expert.

7

Discovering Offensive Patterns in Volleyball Data

Like in soccer and all other dynamic sports, automatically detecting and understanding tactics in volleyball is a challenging problem. Optical tracking provides sufficient context about the game state, which enables to gain insights into tactics and strategies that go beyond simple descriptive statistics. Volleyball tactics comprise complex interactions among multiple players that evolve across both time and space. Hence, strategy detection must consider and evaluate a huge number of possible spatio-temporal movement patterns.

This chapter addresses the task of automatically discovering attacking strategies in volleyball matches based on camera-tracking data of the players and the ball. In particular, this chapter considers data that were collected during the final matches from the 2014 FIVB Volleyball World Championships.

Contributions of this Chapter

The first contribution of this chapter is an inductive-logic-programming approach that automatically discovers offensive patterns in spatio-temporal volleyball match data. More specifically, the approach aims to discover patterns that occur frequently in won rallies and infrequently in lost rallies. In addition,

the approach also aims to find patterns that are used by one team in a particular match but not the opposing team.

The second contribution is an analysis of both the men's and women's final match at the 2014 FIVB Volleyball World Championships. The analysis demonstrates that the proposed approach is able to discover spatio-temporal patterns on multiple different levels of granularity that characterize successful attacking play in professional volleyball matches.

The content of this chapter is based on the following publication:

Jan Van Haaren, Horesh Ben Shitrit, Jesse Davis, and Pascal Fua (2016). "Analyzing Volleyball Match Data from the 2014 World Championships Using Machine Learning Techniques". In: *Proceedings of the Twenty-Second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016; San Francisco, California, United States; 13-17 August 2016)*

Structure of this Chapter

Section 7.1 formally defines the addressed task. **Section 7.2** provides the necessary background on volleyball. **Section 7.3** describes the spatio-temporal dataset. **Section 7.4** introduces the proposed approach. **Section 7.5** presents the analysis of both the men's and women's final match at the 2014 FIVB Volleyball World Championships. **Section 7.6** provides the conclusions.

7.1 Problem Description

In this chapter, we consider the following two tasks:

Task 1: Identify a team's attacking patterns in volleyball matches that occur frequently in won rallies and infrequently in lost rallies.

Task 2: Identify attacking patterns in a volleyball match that are used by one team but not the opposing team.

In contrast to most existing approaches, we attempt to identify patterns that account for both spatial and temporal aspects of the game. That is, we want to model partial or complete configurations of players' positions on the court as

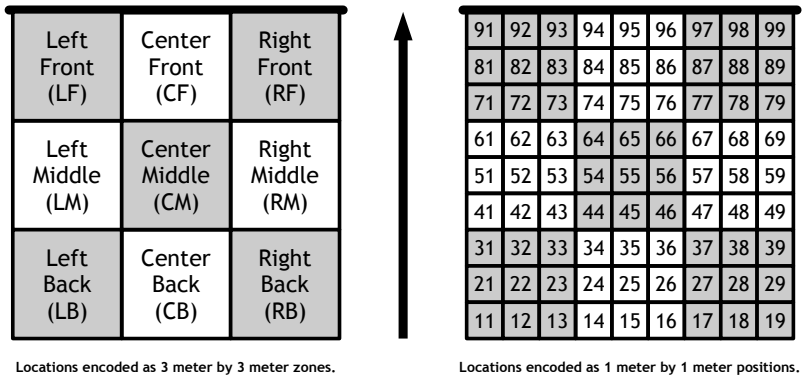


Figure 7.1: The division of the pitch in higher-level zones (left) and lower-level positions (right). The direction of play is from bottom to top, where the thick line at the top represents the net.

well as how play evolves over time. To illustrate this, consider the following simple pattern that was automatically discovered by our approach:

```
IF player #13 performs the dig  AND NEXT
   player #1 performs the set
       in the front center zone AND NEXT
   player #8 performs the spike
       in position 81 of the court
THEN the attack is likely to be successful.
```

This pattern is temporal as the dig occurs first, the set second, and the spike third. The pattern is spatial as it states the location on the court where the set and spike occur. Figure 7.1 shows a description of the locations.

In order to automatically identify patterns like the one above, we use an approach based on relational-learning techniques. As much as possible, we attempt to employ a data-driven approach that can automatically determine which players and characteristics of the game state are relevant to the strategy and should be included in the patterns.

7.2 Background on Volleyball

Volleyball (FIVB 2015) is a ball sport that is played by two teams of six players each. A volleyball court is 18 meters (59 feet) long and 9 meters (29.5 feet) wide. Each team occupies one half of the court, which is 9 meters by 9 meters. The halves are separated by a net whose top is 2.43 meters above the floor in men's competitions and 2.24 meters in women's competitions. The overall goal is to score points by grounding the ball on the opponent's court.

Volleyball matches are won by the first team to win three sets. A set is won by the first team to score 25 points and lead by two points. However, a possible fifth set is typically played to 15 points only. Each set consists of rallies and either of both teams is awarded a point at the end of each rally. A rally starts by serving the ball from behind the back-line over the net into the opponent's court. The opponent may touch the ball up to three times to prevent it from hitting the court and to get the ball back over the net. A rally ends when a team makes either a kill by grounding the ball on the opponent's court or a foul.

The players follow a rotation scheme, where they must rotate one time in clockwise direction after their team wins the serve. Nevertheless, players do have different roles in the team and are free to move within their half of the court after the serve. Typically, the best offensive players will move towards the net, while the best defensive players will move to the back of the court. Although the rotation scheme imposes some restrictions, this tactical freedom allows teams to adopt a wide variety of match strategies. Therefore, volleyball players need to master the following six basic skills:

Serve The serve is the skill of moving the ball from behind the back-line into the opponent's court. While many different types of serves are used, the most popular type is the jump serve, where the server first tosses the ball high in the air and then jumps to hit it.

Dig The dig is the defensive skill of preventing the ball from hitting the court when the ball is nearly touching the floor after a serve or attack from the opponent. The dig is a reflex-based skill which often requires a player to dive towards the ball.

Pass The pass is very similar to the dig. However, in addition to preventing the ball from hitting the court, this defensive skill also involves moving the ball towards a team mate that is well-placed to set up an attack.

Set The set is the offensive skill of pushing the ball into the air such that a team mate can hit it into the opponent's court. The setter, who is the player performing the set, coordinates the offensive play of the team by deciding who will eventually attack the ball.

Spike The spike is the offensive skill of hitting the ball such that the opponent cannot prevent it from touching their court. The spiker, who is the player performing the spike, first makes a few steps and then jumps to swing at the ball.

Block The block is the skill of stopping or altering an opponent's attack by players standing at the net. An offensive block aims at keeping the ball into the opponent's court, while a defensive block aims at getting the ball under control by slowing it down.

Typically, the dig is the first contact with the ball, the set the second, and the spike the third. In this chapter, we will focus on these three skills.

7.3 Dataset

The data were collected at the FIVB Volleyball World Championships finals that were held in Poland[1] and Italy[2] in 2014. PlayfulVision[3] recorded several men's and women's matches, including the final matches of both tournaments. They first captured each match using 8 video cameras placed at different angles at 30 frames per second and then used their ball and player tracking code to automatically determine the locations of the players and the ball in each frame. Furthermore, a human manually annotated each frame with the skills performed by the players (i.e., serve, dig, set, spike, and block) for both finals. The dataset does not distinguish between digs and passes.

In this chapter, we focus on the final matches of the world championships for which both the tracking information and the annotations are available. Table 7.1 shows the number of sets, rallies, and attacks in both matches as well as relevant statistics for each team such as the number of won rallies and how often they performed each of the five considered skills.

[1] http://poland2014.fivb.org/en
[2] http://italy2014.fivb.org/en
[3] PlayfulVision Volleyball Tracking System: http://www.volleyballtracking.com.

		Men's final		Women's final	
		Brazil	**Poland**	**China**	**USA**
Sets	Total sets	4	4	4	4
	Won sets	1	3	1	3
	Lost sets	3	1	3	1
Rallies	Total rallies	185	185	188	188
	Won rallies	92	93	94	94
	Lost rallies	93	92	94	94
Attacks	Total attacks	246	248	300	306
Skills	Total serves	86	89	95	93
	Total digs	124	120	167	173
	Total sets	105	106	145	154
	Total spikes	112	112	148	156
	Total blocks	39	42	51	52

Table 7.1: The statistics for the men's and women's final match at the 2014 FIVB Volleyball World Championships.

We divide each rally into a series of attacks, where each attack consists of a sequence of consecutive skills performed by the same team. In this work, we only consider attacks that involve each of a dig, a set, and a spike. Based on this preprocessing, we construct positive and negative examples for four settings on each task. In task 1, the positive examples are a team's attacks that result in a point, while the negatives are all the team's other attacks. In task 2, the positives are the attacks from one team (e.g., Brazil), while the negatives are the attacks from the opponent (e.g., Poland). Table 7.2 lists the number of positive and negative examples for each setup.

To represent an attack, we take a snapshot of the pitch configuration at the time of each skill. Each snapshot is described by the performed skill (i.e., dig, set or spike) as well as information about the location of each player and the ball. Each snapshot describes the locations on two levels of granularity: high-level zones (see Figure 7.1 left) and lower-level positions (see Figure 7.1 right).

		Men's final		Women's final	
		Brazil	**Poland**	**China**	**USA**
Task 1	Positive examples	51	49	55	61
	Negative examples	18	17	25	36
Task 2	Positive examples	96	95	132	144
	Negative examples	95	96	144	132

Table 7.2: The number of positive and negative examples for each task.

7.4 Methodology

Pattern mining typically focuses on finding patterns that occur frequently in the data. Our problem has several other important characteristics that we must account for and which distinguish it from standard pattern mining. First, each of our tasks requires differentiating between two classes of examples, either successful and unsuccessful attacks or attacks done by two different teams. Second, we want to be able to simultaneously reason about multiple different levels of granularity in the data. For example, we may want to represent the court with both high-level zones and lower-level positions as illustrated in Figure 7.1. Furthermore, we would like to be able to discover patterns that involve specific players in a specific position as well as patterns that involve any player in a specific position. Third, this problem is inherently relational and it is crucial to find patterns that account for relationships such as changes over time. For example, we may want to know how the configuration of players changes between a dig and a set. Finally, we have specific knowledge about volleyball and we would like to be able to incorporate it into the pattern mining process.

Based on the above requirements, we pursue an approach based on inductive logic programming (ILP, Muggleton and De Raedt 1994). ILP is a relational-learning approach that permits modeling multiple granularities, capturing relationships, and incorporating background knowledge. The goal of ILP is to learn a model that distinguishes positive examples from negative examples. The model is a set of IF-THEN rules. The IF portion of a rule is a set of conditions and the THEN portion has an outcome. If all the conditions in the IF portion are met, then the outcome can be expected with a certain probability. An advantage of rules of this form is that they are easy for domain experts to interpret.

However, we do have to make a number of modifications to the standard ILP setup to adapt it to our needs. ILP is traditionally used for learning classifiers, and hence it has a preference for smaller models. That is, models with as few rules as possible with each rule being as short as possible. In contrast, for knowledge discovery we want to find all interesting rules. Furthermore, all other things being equal, we would prefer a detailed pattern that gave as much information about the strategy (e.g., assigns positions and actions to as many players as possible). Thus, unlike model learning, we tend to prefer more specific or detailed patterns to more general ones. To address these problems, we use a two-step process involving pattern generation and pattern post processing.

Pattern Generation

We use the well-known and publicly available Aleph ILP system (Srinivasan 2001) to generate rules. Depending on the task, we define a set of positive and negative examples as described in the Data section. Aleph constructs one rule at a time, with the goal of discovering a set of conditions such that the condition applies to as many positive and as few negative examples as possible. In other words, it tries to maximize the precision of the rule or the conditional probability that the outcome is true given that the condition is true.

Instead of learning one rule at a time, we define a set of criteria and attempt to identify rules that satisfy the criteria. We consider patterns of up to size 15. Each pattern must apply to at least five examples (i.e., a frequency threshold). Each pattern must have a precision of at least 25%. Given the size of the search space (e.g., there are on the order of 10^{22} patterns of size 15 alone), an exhaustive approach is infeasible so we employ a beam search. We collect all rules found by the beam search that meet these conditions.

When constructing a pattern, the learning algorithm can make use of both levels of granularity for the location and it automatically decides, based on the data, which to use. In fact, a single pattern can simultaneously make use of both granularities. The patterns can refer to specific players (e.g., player 1 is in the front center zone) or generic ones (e.g., a player is in the front center zone). Referring to generic players allows us to find patterns that generalize across different lineups for a team as teams may substitute during the match.

Pattern Post-processing

Even with our constraints, a huge number of patterns were generated. For task 1, there were 53,212 patterns generated for Brazil, 79,907 for Poland, 51,593 for China, and 77,509 for the USA. For task 2, there were 190,956 patterns generated for Brazil, 197,467 for Poland, 269,753 for China, and 327,731 for the USA. We post processed the patterns and only retained those that included all three skills (dig, set, and spike) we were interested in. This resulted in 1,527 patterns for Brazil, 1,301 for Poland, 242 for China, and 2,037 for the USA in task 1. For task 2, there were 65,484 patterns generated for Brazil, 41,763 for Poland, 73,011 for China, and 84,639 for the USA.

Then, we ranked the patterns by the number of specific players and locations the patterns contained. This enforces a preference for longer, more specific patterns. We broke ties by considering pattern coverage (number of positives the pattern applies to), pattern precision, and pattern length.

7.5 Results and Discussion

We present the top-ranked pattern for each of the eight setups we consider. Most of these patterns capture the same offensive strategy, which involves attacking over one side of the court. The patterns do have some small variations, such as about whether the attack is on the left or right side. This attacking pattern is a well-known volleyball strategy. Next, we discuss the top-ranked pattern in each setup in more detail.

Task 1: Which Offensive Strategy is Most Successful for Each Team?

Figures 7.2 through 7.5 visualize the offensive patterns that occur in successful attacks but not unsuccessful ones. In all figures, numbers represent specific players, the capital letter X represents the location of the ball, and other capital letters (A, B, and C) represent generic players. That is, they denote the presence of a player in that location, but do not specify which player is there, and hence account for the fact that teams can substitute players. The net is shown by a thick line at the top of the image.

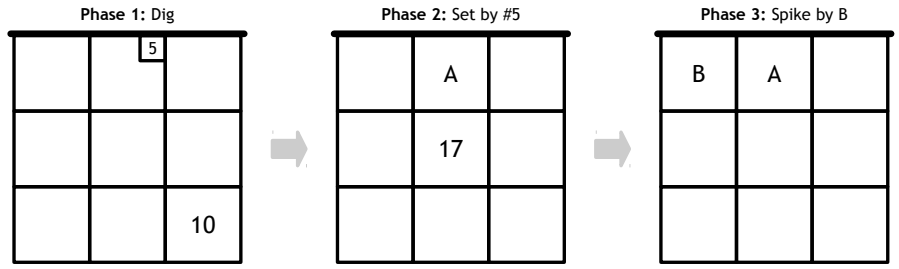


Figure 7.2: A frequent successful offensive pattern by Poland in the men's final. The setter, who is the player with jersey number 5, is already close to the net at the time of the dig. He sets the ball to the front left zone, where another player B spikes it. Here, player B could denote any other Polish player.

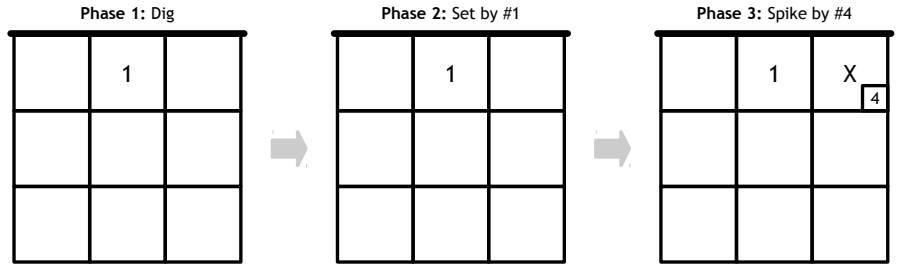


Figure 7.3: A frequent successful offensive pattern by Brazil in the men's final. The setter, who is the player with jersey number 1, is already close to the net at the time of the dig. He sets the ball from the front center zone to the back right corner of the front right zone, where the player with jersey number 4 spikes it. X denotes the location of the ball.

Figure 7.2 shows the top-ranked pattern for the Poland men's team. The setter, who is the player with jersey number 5, is already close to the net at the time of the dig. He sets the ball to the front left zone, where another player B spikes it. The pattern covers five successful attacks and no unsuccessful ones. Looking at the location of the spike, the Polish team had 36 successful (including the five covered by this pattern) and 16 unsuccessful spikes in the front left zone.

Figure 7.3 shows the top-ranked pattern for the Brazil men's team. This pattern is very similar to the one for the Polish team, but with two important exceptions. First, the spike occurs on the right side of the court. Second, the spike occurs in a very specific location, at the back right corner of the front right zone.

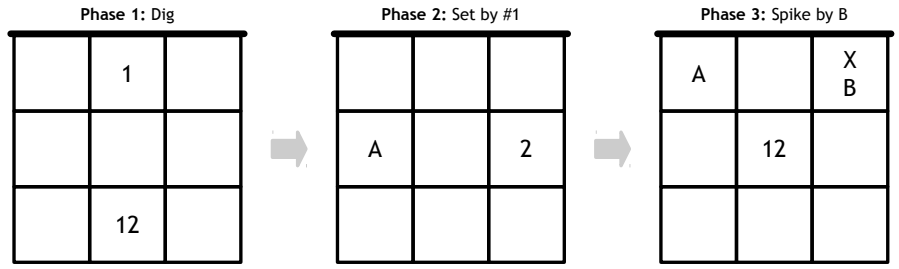


Figure 7.4: A frequent successful offensive pattern by the USA in the women's final. Player 1, who is the setter, is in the front center at the time of the dig. Player 1 sets the ball to the front left where player B spikes it. A player denoted A moves from the left center to the left front between the dig and the set. It appears that the setter could set to either side for a spike. X denotes the location of the ball.

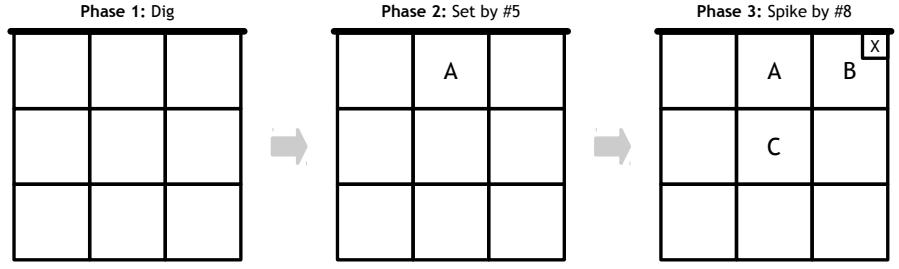


Figure 7.5: A frequent successful offensive pattern by China in the women's final. In this pattern, player 5 sets the ball and it is spiked by player 8 in the front right corner of the court. X denotes the location of the ball.

The pattern covers five successful attacks and no unsuccessful ones. Here, the specific location is important to the pattern as, in the entire front right zone, Brazil had 22 successful attacks (including the five covered by this pattern) and 13 unsuccessful attacks. In terms of the specific position where player 4 was located, Brazil attempted 11 spikes of which nine were successful. Thus the team was much more successful in this location than in the zone in general. The Polish team was less successful from this specific location, attempting 15 spikes of which 10 were successful. Also, in contrast to the women's final, the USA and China only attempted four spikes in aggregate in this specific location and none of them were successful.

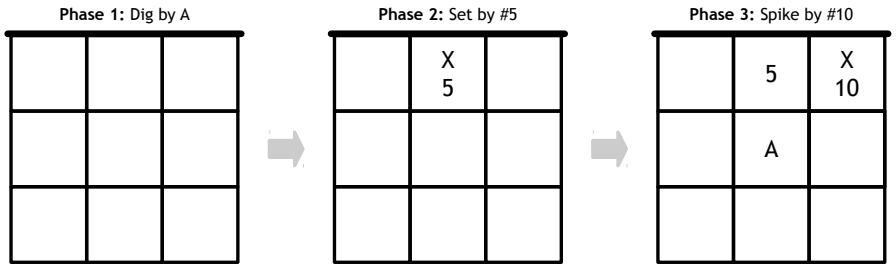


Figure 7.6: A frequent offensive pattern by Poland that Brazil did not frequently use in the men's final. Player with jersey number 5 sets the ball from the front center to the front right where it is spiked by player number 10. X denotes the location of the ball.

Figure 7.4 shows the top-ranked pattern for the USA women's team. This again illustrates an attack from the side of the court at the front. Given that player B spikes the ball and the ball is located in the front left zone, that is the location where the spike was performed. Notice that a player denoted A moves from the left middle to the left front between the dig and the set. Thus, the setter could have set to either side for the spike. The pattern covers five successful attacks and no unsuccessful ones. The USA attempted 56 spikes in the front right zone of which 37 were successful (including the five covered by this pattern) and 19 were unsuccessful.

Figure 7.5 shows the top-ranked pattern for the China women's team. This pattern is less specific than the previous three, but it still shows the same general scheme of attacking from the side of the court on the front. Given that the ball is in the front rightmost corner next to the net at the time of the spike, we can infer that this is where player 8 is located. The pattern covers five successful attacks and no unsuccessful ones. Over all spikes in this specific position, China had eight successful spikes (including the five covered by this pattern) and two unsuccessful ones.

Task 2: Which Common Offensive Strategy Distinguishes a Team from Its Opponent?

Figures 7.6 through 7.9 show the offensive patterns that distinguish between two teams playing against each other. The patterns are illustrated in the same

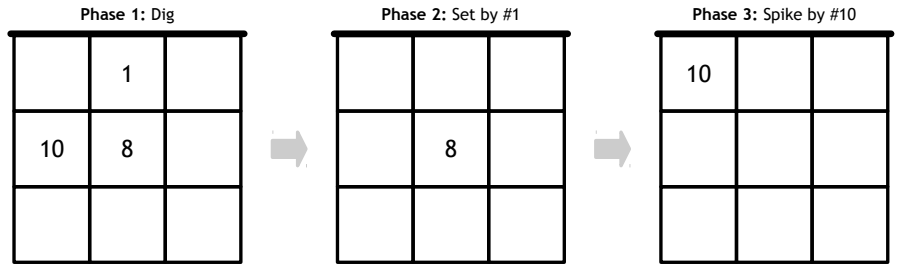


Figure 7.7: A frequent offensive pattern by Brazil that Poland did not frequently use in the men's final. The player with jersey number 1 is located in the front center at the time of the dig. He sets the ball from the front center to the front left. Player 10 moved from the middle left zone at the time of the dig to the front left zone, where he spiked the ball.

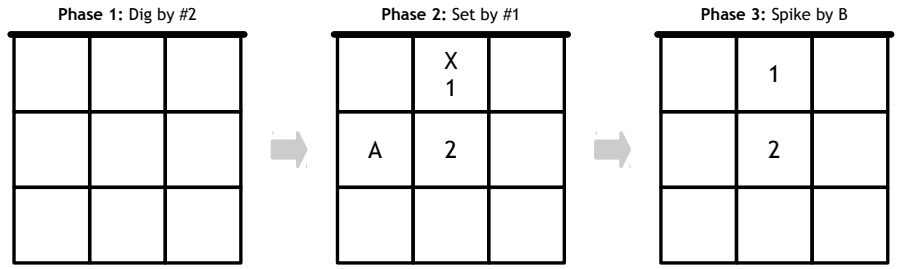


Figure 7.8: A frequent offensive pattern by the USA that China did not frequently use in the women's final. The ball is set by the player with jersey number 1 in the front center zone. Afterwards, another player spikes the ball. X denotes the location of the ball.

manner as for Task 1.

Figure 7.6 shows the top-ranked pattern employed by the Poland men's team that is not used by the Brazilian team in their match. Figure 7.7 shows the top-ranked pattern for the Brazil men's team that is not used by the Polish team. These patterns are quite similar, with a set in the middle and a spike by player 10 on the outside in the front. The difference is that the Polish number 10 attacks on the right and the Brazilian number 10 on the left. One possible explanation is that these players have different dominant hands. Typically in volleyball the spiker wants his dominant hand closest to where the ball is coming from (that is, right-handed players want to spike from the front left, and left-handed players

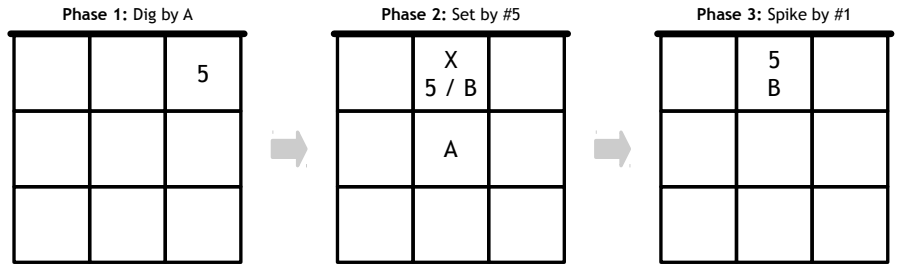


Figure 7.9: A frequent offensive pattern by China that the USA did not frequently use in the women's final. The ball is set by player 5 in the front center zone. Afterwards, player 1 spikes the ball. X denotes the location of the ball.

want to spike from the front right). We could not verify this hypothesis.

Figure 7.8 shows the top-ranked pattern for the USA women's team that is not used by the Chinese team. Figure 7.9 shows the top-ranked pattern for the China women's team that is not used by the USA team. These patterns are slightly less informative as they do not indicate the location of the spike. All the patterns tell is that the ball is set in the front center zone.

How Do Men's and Women's Volleyball Compare to Each Other?

Another natural question to ask is how do the men's and women's game compare to each other. The statistics in Table 7.1 show that there are some commonalities between them. Namely, the number of rallies is roughly the same in both finals. However, the women's match features many more attacks than the men's final. Furthermore, the number of digs, set, spikes, and blocks is much higher in the women's final than in the men's final. This is most likely due to the faster pace of men's volleyball, which makes it harder to gain control of the ball after an attack from the opponent.

7.6 Conclusions

This chapter investigated the task of automatically discovering recurring patterns in successful offensive patterns in volleyball matches and presented an inductive-logic-programming approach that addresses this task.

The proposed approach views a volleyball match as a sequence of events and performs the following three steps. First, the approach splits each match into a set of subrallies, which are subsequences of events within a rally. Second, it labels the subrallies that lead to a point as positive and all others as negative. Third, the approach learns rules describing patterns that appear more often in subrallies labeled positive than in subrallies labeled negative.

Our experimental evaluation on both the men's and women's final match of the 2014 FIVB Volleyball World Championships demonstrates that the proposed approach is able to discover spatio-temporal patterns on multiple different levels of granularity that characterize successful attacking play in volleyball.

In the future, it may be possible to improve the performance of the approach by explicitly taking into account the differences in playing style of the different opponents. The proposed approach might also benefit from a more sophisticated, domain-driven procedure to assign the labels to the subrallies.

8

Conclusions

This chapter summarizes the presented contributions and discusses promising avenues for future research on the topics that this dissertation encompasses.

8.1 Summary

Most traditional machine-learning algorithms cannot be applied directly to important real-world problems as they rely on assumptions that do not hold in practice. One assumption is that the data are represented in a simple, tabular format, while most real-world domains exhibit a complex, relational structure involving different types of entities and relationships. Another assumption is that plenty of data is available to learn an accurate predictive model, while real-world domains often exhibit a shortage of high-quality data.

This dissertation aims to overcome the limitations of traditional machine-learning techniques by presenting novel algorithms in the field of statistical relational learning, which can deal with uncertainty and complexly-structured data in a natural way. Furthermore, this dissertation also applies existing relational-learning techniques to spatio-temporal sports data, which exhibit the many challenges that real-world applications pose.

Contributions

This dissertation presents five main contributions. The first three contributions are relevant to the domain of statistical relational learning, while the remaining two contributions are relevant to the field of sports analytics. We identify the following five main contributions:

1. A novel algorithm for learning Markov random fields from data
2. A novel algorithm for learning tractable Markov logic networks from data
3. A novel framework and approximate algorithm for performing deep-transfer learning in relational domains
4. An inductive-logic-programming approach for automatically discovering offensive strategies in spatio-temporal soccer match data
5. An inductive-logic-programming approach for automatically discovering offensive patterns in spatio-temporal volleyball match data

Structure Learning of Markov Random Fields

The first main contribution is a novel algorithm called GSSL (**G**enerate **S**elect **S**tructure **L**earning) for automatically learning Markov random fields from data. The algorithm views structure learning as a feature-induction problem. GSSL first quickly generates a set of candidate features and then selects a subset of those features to include in the final Markov random field.

The proposed algorithm combines some of the benefits of existing structure-learning approaches. Unlike search-based approaches, GSSL avoids the computational cost of performing parameter learning to evaluate each candidate feature. Unlike local-model-based approaches, GSSL avoids the burden of learning the local models, which can quickly become computationally expensive in domains containing a large number of variables or examples.

An extensive empirical evaluation on 20 real-world datasets demonstrates the advantages of the proposed algorithm. Striking in its simplicity, GSSL learns more accurate models and is much faster than the baseline approaches.

Lifted Structure Learning of Markov Logic Networks

The second main contribution is a novel algorithm called LSL (**L**ifted **S**tructure **L**earning) for automatically learning tractable Markov logic networks from data. The algorithm views structure learning as performing a greedy search in a space of candidate models. LSL first generates a set of candidate formulas and then iteratively adds formulas to an initially empty Markov logic network. To enforce the tractability of the final model, LSL discards all candidate models that do not allow lifted inference and therefore are considered intractable.

The proposed algorithm leverages techniques from lifted inference to optimize the exact training-set log-likelihood and to learn models that are guaranteed to support certain types of queries efficiently. In contrast, standard structure-learning algorithms resort to approximate techniques that lead to intractable models and thus inaccurate predictions.

An extensive empirical evaluation on three real-world datasets demonstrates the advantages of the proposed algorithm. LSL learns more accurate models than other tractable learners in terms of test-set log-likelihood as well as area under the precision-recall curve and conditional log-likelihood on prediction tasks. Surprisingly, LSL even outperforms standard algorithms that learn intractable models on prediction tasks, suggesting that tractable models are a powerful hypothesis space that is sufficient for many standard problems.

Deep Transfer Learning in Relational Domains

The third main contribution is a novel framework called TODTLER (**T**wo-**O**rder-**D**eep **T**ransfer **Le**a**r**ning) for automatically performing deep-transfer learning in relational domains. The framework views knowledge transfer as the process of first learning a declarative bias in a source domain, where the data are ample, and then applying that bias to a target domain, where the data are scarce, to improve the learning process. The framework exploits the observation that many domains share structural regularities although their high-level descriptions (e.g., properties, entities, and relationships) can be entirely different.

The concrete implementation of the TODTLER framework performs deep-transfer learning, where the source and target domains can consist of entirely different sets of entities and relationships, in the context of Markov logic networks. TODTLER represents the transferable knowledge as a distribution

over domain-independent second-order templates, which give rise to first-order Markov logic formulas when applied to a particular domain.

An extensive empirical evaluation on three real-world datasets demonstrates the advantages of the proposed framework. TODTLER learns more accurate models than the state-of-the-art deep-transfer learning algorithm as well as the state-of-the-art inductive-learning algorithm. In addition to learning more accurate models, TODTLER is also much faster and can transfer a broader range of knowledge than the existing deep-transfer-learning algorithms.

Discovering Offensive Patterns and Strategies in Sports Data

The fourth main contribution is an inductive-logic-programming-approach that automatically discovers offensive strategies in spatio-temporal soccer match data. The approach views a soccer match as a long stream of events. First, it identifies subsequences of related events, which are called phases. Second, it discovers recurring patterns in the identified phases. The approach aims to discover patterns that frequently occur in phases leading to a goal attempt.

An empirical study on a large volume of matches from a professional soccer club demonstrates that the proposed approach is able to automatically discover interesting and relevant offensive strategies in soccer match data.

The fifth main contribution is an inductive-logic-programming approach that automatically discovers offensive patterns in spatio-temporal volleyball match data. The approach views a volleyball match as a set of sequences of events, where each sequence corresponds to a rally. First, it identifies subsequences of events in each rally, which are called subrallies. Second, it discovers recurring patterns in the identified subrallies. The approach aims to discover patterns that frequently occur in won rallies and infrequently in lost rallies.

An analysis of both the men's and women's final match at the 2014 FIVB Volleyball World Championships demonstrates that the proposed approach is able to discover offensive patterns in volleyball match data.

The crucial difference between the fourth and fifth contribution is in the way the raw tracking data are transformed into a logical representation. Soccer and volleyball are inherently different sports and our logical representation needs to account for at least the following mutual differences:

1. Soccer players can freely move around the pitch and often directly interact with their opponents. Volleyball players are separated by a net.
2. Soccer pitches are much larger than volleyball pitches. Hence, analyzing volleyball matches requires a more fine-grained pitch representation than analyzing soccer matches, where minor movements are less important.
3. Soccer players have fixed positions on the pitch. Volleyball players rotate one position to the left each time they gain the right to serve.
4. Soccer events can happen in almost any order. Volleyball events tend to follow the dig-set-spike pattern in most rallies.

8.2 Discussion, Perspectives, and Future Work

The contributions presented in this dissertation provide valuable advances and insights concerning the applicability of statistical-relational-learning techniques in real-world domains, which are often characterized by complex relational data and a shortage of high-quality training data. Nevertheless, many open questions remain and should be addressed to enable a broader adoption of machine-learning techniques in challenging but important real-world applications.

This section presents several possible avenues for future work with respect to structure learning and transfer learning as well as sports analytics.

8.2.1 Structure Learning

Although the structure-learning tasks in Markov random fields and Markov logic networks somewhat differ, they follow a similar paradigm and pose similar challenges. This dissertation views structure learning as a task of first generating candidate features and then selecting a subset of the features to include in the final model. However, evaluating candidate features requires parameter learning and thus inference, which can quickly become computationally expensive in large domains. Hence, the challenge is to reduce the space of possible models by generating the candidate features in a clever way.

The lack of a general understanding of what makes a good feature, complicates the structure-learning task further. The main reason is that a feature's quality does not only depend on the feature itself but also on the other features in the

model. If candidate features $V_0 = 1 \wedge V_1 = 1$ and $V_0 = 1 \wedge V_1 = 1 \wedge V_2 = 1$ have similar support in the data, does adding the latter feature yield a better model if the former feature is already present? Since the latter feature is likely to improve the model's accuracy, the underlying question is whether a possibly marginal improvement in accuracy justifies an increase in the model's complexity.

Markov Random Fields

In the context of Markov random fields, generating a more diverse set of candidate features is a promising research direction to improve the structure-learning process. A disadvantage of GSSL's naive feature-generation approach is that certain regions of the search space are likely to be explored more than others, which can lead to homogeneous candidate-feature sets. A similar problem arises in clique mining, where diverse cliques are often more interesting than largely overlapping cliques. Bogdanov et al. (2013) address this problem by proposing a scalable algorithm for mining diverse cliques in weighted graphs. The feature-generation problem can easily be casted into this setting by constructing a graph where the nodes represent the variables and the edge weights denote how often the variables occur together in the training data.

Markov Logic Networks

In the context of Markov logic networks, leveraging the semantics of the predicates is a promising research direction to improve the structure-learning process. A disadvantage of LSL's and TODTLER's naive formula-generation approach, which simply enumerates all valid formulas up to a certain number of literals and logical variables, is that many generated formulas intuitively make little sense. A possible solution to this problem is leveraging domain knowledge to restrict the number of literals and logical variables per type instead of the total number of literals and logical variables appearing in the formula.

Furthermore, an alternative way of speeding up the lifted-structure-learning approach is to incrementally compile the candidate models into circuits. Currently, each candidate model is compiled from scratch, yielding many unnecessary computations in each structure-learning iteration.

8.2.2 Transfer Learning

Current deep-transfer learning approaches, including the presented TODTLER framework, assume that an appropriate source domain is available when addressing a task in a particular target domain. However, in the real world, it is likely that several different candidate-source domains are available. Hence, the task is either to select the most appropriate source domain or to combine the knowledge acquired from several different appropriate source domains.

One possible research direction is developing a source-selection algorithm that identifies the most appropriate source domain from a set of candidate-source domains given a target domain. This problem can be posed as a clustering problem, where the distance between two domains is defined in terms of their meta features. Possibly relevant meta features include the number of distinct types of entities and relationships in the domain as well as the average number of relationships that each entity is involved in.

Another possible research direction is adapting the proposed TODTLER framework to support multi-source transfer learning. The natural way of doing this is assigning a weight to each domain and taking this weight into account when computing a formula's probability of inclusion in the target domain model. More specifically, in Algorithm 5, this would correspond to performing lines 5 to 8 for each source domain and taking the weight into account when computing the approximate sampling probability for each template on line 9.

8.2.3 Sports Analytics

Sports analytics is a broad field that encompasses many different tasks and disciplines. This dissertation focuses on techniques for automatically discovering the strategies used by a particular team. Due to a lack of powerful computational methods that can operate on complexly-structured spatio-temporal sports data, strategy discovery is virtually unexplored to date.

One possible research direction is developing a dedicated language to describe sports tactics and strategies. Sports scientists and machine-learning researchers currently speak different languages and a dedicated language that both sides understand would bridge this gap. While powerful enough to describe a wide range of tactics and strategies, the language should also be easily translatable into a more powerful representation such as first-order logic. For example,

in soccer, the language should have constructs for common concepts such as overlapping defenders and wingers cutting inside. The language would ideally be developed in close collaboration with sports scientists and practitioners.

Another possible research direction is developing a relational expected-goals model for soccer. The most-widespread advanced performance metric in soccer is the expected-goals value of a goal attempt. Given that luck plays an important role in soccer, this metric aims to measure the quality of goal attempts in an objective manner. Current expected-goals models largely ignore the context of goal attempts by only considering simple characteristics such as the distance and angle to the goal. However, given that relational-learning techniques can deal with complex sports data in a natural way, they are able to take the context into account as well and therefore are likely to learn more accurate models.

Bibliography

Benjamin Alamar (2013). *Sports Analytics: A Guide for Coaches, Managers, and Other Decision Makers*. Columbia University Press (cited on p. 5).

Galen Andrew and Jianfeng Gao (2007). "Scalable Training of L_1-Regularized Log-Linear Models". In: *Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML 2007; Corvallis, Oregon, United States; 20-24 June 2007)*. ACM Press, pages 33–40 (cited on p. 28).

Bikramjit Banerjee, Yaxin Liu, and Michael Youngblood, editors (2006). *ICML 2006 Workshop on Structural Knowledge Transfer for Machine Learning* (cited on p. 56).

Jonathan Baxter, Rich Caruana, Tom Mitchell, Lorien Pratt, Daniel Silver, and Sebastian Thrun, editors (1995). *NIPS 1995 Workshop on Learning to Learn: Knowledge Consolidation and Transfer in Inductive Systems* (cited on p. 56).

Julian Besag (1975). "Statistical Analysis of Non-Lattice Data". In: *Journal of the Royal Statistical Society. Series D. (The Statistician)* 24(3), pages 179–195 (cited on p. 13).

Carl Bialik (2014a). *Statkeepers Call the Shots, but They Can't Agree on Them*. URL: https://www.fivethirtyeight.com/features/statkeepers-call-the-shots-but-they-cant-agree-on-them (cited on p. 2).

Carl Bialik (2014b). *The People Tracking Every Touch, Pass and Tackle in the World Cup*. URL: https://www.fivethirtyeight.com/features/the-people-

tracking-every-touch-pass-and-tackle-in-the-world-cup (cited on p. 2).

Alina Bialkowski, Patrick Lucey, Peter Carr, Yisong Yue, Sridha Sridharan, and Iain Matthews (2014). "Identifying Team Style in Soccer Using Formations Learned from Spatiotemporal Tracking Data". In: *Proceedings of the ICDM 2014 Workshop on Spatial and Spatiotemporal Data Mining*, pages 9–14 (cited on p. 79).

Petko Bogdanov, Ben Baumer, Prithwish Basu, Amotz Bar-Noy, and Ambuj Singh (2013). "As Strong as the Weakest Link: Mining Diverse Cliques in Weighted Graphs". In: *Proceedings of the 2013 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2013; Prague, Czech Republic; 23-27 September 2013)*. Springer, pages 525–540 (cited on p. 112).

Kendrick Boyd, Vitor Santos Costa, Jesse Davis, and David Page (2012). "Unachievable Region in Precision-Recall Space and Its Effect on Empirical Evaluation". In: *Proceedings of the Twenty-Ninth International Conference on Machine Learning (ICML 2012; Edinburgh, Scotland; 26 June - 1 July 2012)*, pages 639–646 (cited on pp. 46, 66).

Bojan Cestnik (1990). "Estimating Probabilities: A Crucial Task in Machine Learning". In: *Proceedings of the Ninth European Conference on Artificial Intelligence (ECAI 1990; Stockholm, Sweden; 6-10 August 1990)*. Volume 90, pages 147–149 (cited on p. 83).

Anton Chechetka and Carlos Guestrin (2007). "Efficient Principled Learning of Thin Junction Trees". In: *Advances in Neural Information Processing Systems 20 (NIPS 2007; Vancouver, British Columbia, Canada; 3-8 December 2007)*, pages 273–280 (cited on pp. 40, 53).

Brian Chen (2016). *Siri, Alexa and Other Virtual Assistants Put to the Test*. URL: http://www.nytimes.com/2016/01/28/technology/personaltech/siri-alexa-and-other-virtual-assistants-put-to-the-test.html (cited on p. 3).

Mark Craven and Sean Slattery (2001). "Relational Learning with Statistical Predicate Invention: Better Models for Hypertext". In: *Machine Learning* 43(1), pages 97–119 (cited on p. 65).

Jesse Davis, Elizabeth Burnside, Ines de Castro Dutra, David Page, and Vitor Santos Costa (2005). "An Integrated Approach to Learning Bayesian Networks of Rules". In: *Proceedings of the Sixteenth European Conference on Machine Learning (ECML 2005; Porto, Portugal; 3-7 October 2005)*, pages 84–95 (cited on p. 65).

Jesse Davis and Pedro Domingos (2009). "Deep Transfer via Second-Order Markov Logic". In: *Proceedings of the Twenty-Sixth International Conference on Machine Learning (ICML 2009; Montreal, Quebec, Canada; 14-18 June 2009)*, pages 217–224 (cited on pp. 19, 64).

Jesse Davis and Pedro Domingos (2010). "Bottom-Up Learning of Markov Network Structure". In: *Proceedings of the Twenty-Seventh International Conference on Machine Learning (ICML 2010; Haifa, Israel; 21-24 June 2010)*. ACM Press, pages 271–278 (cited on pp. 14, 21, 27).

Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton (2008). *Probabilistic Inductive Logic Programming: Theory and Applications*. Springer (cited on pp. 2, 3).

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty (1997). "Inducing Features of Random Fields". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, pages 380–392 (cited on pp. 12, 14, 21).

Pedro Domingos and Austin Webb (2012). "Tractable Markov Logic". In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012; Toronto, Ontario, Canada; 22-26 July 2012)* (cited on p. 54).

Sašo Džeroski and Nada Lavrač (2001). *An Introduction to Inductive Logic Programming* (cited on p. 9).

Brian Falkenhainer, Kenneth Forbus, and Dedre Gentner (1989). "The Structure-Mapping Engine: Algorithm and Examples". In: *Artificial Intelligence* 41(1), pages 1–63 (cited on p. 19).

FIVB (2015). *Official Volleyball Rules*. URL: http://www.fivb.org/EN/Refereeing-Rules/documents/FIVB_Volleyball_Rules_2015-2016_EN_V3_20150205.pdf (cited on p. 94).

Lise Getoor and Ben Taskar, editors (2007). *An Introduction to Statistical Relational Learning*. MIT Press (cited on pp. 2, 3).

Wally Gilks, Sylvia Richardson, and David Spiegelhalter (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall (cited on p. 13).

Laszlo Gyarmati, Haewoon Kwak, and Pablo Rodriguez (2014). "Searching for a Unique Style in Soccer". In: *Proceedings of the KDD 2014 Workshop on Large-Scale Sports Analytics (LSSA 2014; New York City, New York, United States; 24 August 2014)* (cited on p. 79).

Franciso Herrera, Cristóbal José Carmona, Pedro González, and María José del Jesus (2011). "An Overview on Subgroup Discovery: Foundations and Applications". In: *Knowledge and Information Systems* 29(3), pages 495–525 (cited on p. 79).

Tuyen Huynh and Raymond Mooney (2008). "Discriminative Structure and Parameter Learning for Markov Logic Networks". In: *Proceedings of the Twenty-*

Fifth International Conference on Machine Learning (ICML 2008; Helsinki, Finland; 5-9 July 2008). ACM Press, pages 416–423 (cited on p. 26).

Tuyen Huynh and Raymond Mooney (2009). "Max-Margin Weight Learning for Markov Logic Networks". In: *Proceedings of the 2009 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2009; Bled, Slovenia; 7-11 September 2009)*, pages 564–579 (cited on p. 17).

Cecilia Kang (2016). *Self-Driving Cars Gain Powerful Ally: The Government*. URL: http://www.nytimes.com/2016/09/20/technology/self-driving-cars-guidelines.html (cited on p. 3).

Kristian Kersting (2012). "Lifted Probabilistic Inference". In: *Proceedings of the Twentieth European Conference on Artificial Intelligence (ECAI 2012; Montpellier, France; 27-31 August 2012)*, pages 33–38 (cited on pp. 17, 39).

Konstantin Knauf and Ulf Brefeld (2014). "Spatio-Temporal Convolution Kernels for Clustering Trajectories". In: *Proceedings of the KDD 2014 Workshop on Large-Scale Sports Analytics (LSSA 2014; New York City, New York, United States; 24 August 2014)* (cited on p. 79).

Arno Knobbe (2004). "Multi-Relational Data Mining". PhD thesis. Utrecht University (cited on p. 79).

Stanley Kok and Pedro Domingos (2005). "Learning the Structure of Markov Logic Networks". In: *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML 2005; Bonn, Germany; 7-11 August 2005)*, pages 441–448 (cited on pp. 18, 46).

Stanley Kok and Pedro Domingos (2010). "Learning Markov Logic Networks Using Structural Motifs". In: *Proceedings of the Twenty-Seventh International Conference on Machine Learning (ICML 2010; Haifa, Israel; 21-24 June 2010)*, pages 551–558 (cited on pp. 18, 64).

Stanley Kok, Marc Sumner, Matthew Richardson, Parag Singla, Hoifung Poon, Daniel Lowd, Jue Wang, Aniruddh Nath, and Pedro Domingos (2010). *The Alchemy System for Statistical Relational AI*. Technical report. Seattle, Washington, United States: Department of Computer Science and Engineering, University of Washington. URL: http://alchemy.cs.washington.edu (cited on pp. 47, 66).

Petra Kralj Novak, Nada Lavrač, and Geoffrey Webb (2009). "Supervised Descriptive Rule Discovery: A Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Mining". In: *Journal of Machine Learning Research* 10, pages 377–403 (cited on p. 79).

Alex Kulesza and Fernando Pereira (2008). "Structured Learning with Approximate Inference". In: *Advances in Neural Information Processing Systems*

20 (NIPS 2007; Vancouver, British Columbia, Canada; 3-8 December 2007). MIT Press, pages 785–792 (cited on p. 13).

Nada Lavrač, Bojan Cestnik, Dragan Gamberger, and Peter Flach (2004). "Decision Support Through Subgroup Discovery: Three Case Studies and the Lessons Learned". In: *Machine Learning* 57(1-2), pages 115–143 (cited on p. 79).

Nada Lavrač, Sašo Džeroski, and Ivan Bratko (1996). "Handling Imperfect Data in Inductive Logic Programming". In: *Proceedings of the Sixth International Workshop on Inductive Logic Programming (ILP 1996; Stockholm, Sweden; 28-30 August 1996)*. Volume 32, pages 48–64 (cited on p. 83).

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep Learning". In: *Nature* 521(7553), pages 436–444 (cited on p. 3).

Su-In Lee, Varun Ganapathi, and Daphne Koller (2007). "Efficient Structure Learning of Markov Networks Using L_1-Regularization". In: *Advances in Neural Information Processing Systems 19 (NIPS 2006; Vancouver, British Columbia, Canada; 4-9 December 2006)*. MIT Press, pages 817–824 (cited on p. 29).

Michael Lewis (2004). *Moneyball: The Art of Winning an Unfair Game*. W. W. Norton & Company (cited on p. 77).

Dong Liu and Jorge Nocedal (1989). "On the Limited Memory BFGS Method for Large Scale Optimization". In: *Mathematical Programming* 45(3), pages 503–528 (cited on p. 47).

Daniel Lowd and Jesse Davis (2010). "Learning Markov Network Structure with Decision Trees". In: *Proceedings of the Tenth IEEE International Conference on Data Mining (ICDM 2010; Sydney, Australia; 14-17 December 2010)*, pages 334–343 (cited on pp. 15, 22, 27, 29).

Daniel Lowd and Pedro Domingos (2007). "Efficient Weight Learning for Markov Logic Networks". In: *Proceedings of the Eleventh European Conference on Principles and Practices of Knowledge Discovery in Databases (PKDD 2007; Warsaw, Poland; 17-21 September 2007)*, pages 200–211 (cited on pp. 17, 47).

Daniel Lowd and Pedro Domingos (2008). "Learning Arithmetic Circuits". In: *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI 2008; Helsinki, Finland; 9-12 July 2008)*, pages 383–392 (cited on p. 54).

Daniel Lowd and Amirmohammad Rooshenas (2013). "Learning Markov Networks with Arithmetic Circuits". In: *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2013; Scottsdale, Arizona, United States; 29 April - 1 May 2013)*, pages 406–414 (cited on p. 54).

Patrick Lucey, Dean Oliver, Peter Carr, Joe Roth, and Iain Matthews (2013). "Assessing Strategy Using Spatio-Temporal Data". In: *Proceedings of the*

Nineteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2013; Chicago, Illinois, United States; 11-14 August 2013), pages 1366–1374 (cited on p. 79).

Andrew McCallum (2003). "Efficiently Inducing Features of Conditional Random Fields". In: *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI 2003; Acapulco, Mexico; 7-10 August 2003)*, pages 403–410 (cited on p. 21).

Hans-Werner Mewes, Dmitrij Frishman, Christian Gruber, Birgitta Geier, Dirk Haase, Andreas Kaps, Kai Lemcke, Gertrud Mannhaupt, Friedhelm Pfeiffer, Christine Schüller, S. Stocker, and B. Weil (2000). "MIPS: A Database for Genomes and Protein Sequences". In: *Nucleic Acids Research* 28(1), pages 37–40 (cited on p. 65).

Lilyana Mihalkova, Tuyen Huynh, and Raymond Mooney (2007). "Mapping and Revising Markov Logic Networks for Transfer Learning". In: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007; Vancouver, British Columbia, Canada; 22-26 July 2007)*, pages 608–614 (cited on pp. 19, 65).

Lilyana Mihalkova and Raymond Mooney (2007). "Bottom-Up Learning of Markov Logic Network Structure". In: *Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML 2007; Corvallis, Oregon, United States; 20-24 June 2007)*, pages 625–632 (cited on pp. 18, 21, 45, 46).

David Moore and Andrea Danyluk (2010). "Deep Transfer as Structure Learning in Markov Logic Networks". In: *Proceedings of the First International Workshop on Statistical Relational AI (StaRAI 2010; Atlanta, Georgia, United States; 12 July 2010)*, pages 52–57 (cited on p. 76).

Stephen Muggleton and Luc De Raedt (1994). "Inductive Logic Programming: Theory and Methods". In: *The Journal of Logic Programming* 19, pages 629–679 (cited on p. 97).

Kevin Murphy, Yair Weiss, and Michael Jordan (1999). "Loopy Belief Propagation for Approximate Inference: An Empirical Study". In: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI 1999; Stockholm, Sweden; 30 July - 1 August 1999)*, pages 467–475 (cited on p. 13).

Christopher Mutschler, Holger Ziekow, and Zbigniew Jerzak (2013). "The DEBS 2013 Grand Challenge". In: *Proceedings of the Seventh International Conference on Distributed Event-based Systems (DEBS 2013; Arlington, Texas, United States; 29 June - 3 July 2013)*, pages 289–294 (cited on p. 79).

Mukund Narasimhan and Jeff Bilmes (2004). "PAC-Learning Bounded Tree-width Graphical Models". In: *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI 2004; Banff, Alberta, Canada; 7-11 July 2004)*, pages 410–417 (cited on p. 53).

Tim Op De Beéck, Arjen Hommersom, Jan Van Haaren, Maarten van der Heijden, Jesse Davis, Peter Lucas, Lucy Overbeek, and Iris Nagtegaal (2015). "Mining Hierarchical Pathology Data Using Inductive Logic Programming". In: *Proceedings of the Fifteenth Conference on Artificial Intelligence in Medicine (AIME 2015; Pavia, Italy; 17-20 June 2015)*, pages 76–85 (cited on pp. 89, 128).

Sinno Jialin Pan and Qiang Yang (2010). "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22(10), pages 1345–1359 (cited on pp. 2, 19).

David Poole (2003). "First-Order Probabilistic Inference". In: *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003; Acapulco, Mexico; 9-15 August 2003)*, pages 985–991 (cited on pp. 17, 39).

Pradeep Ravikumar, Martin Wainwright, and John Lafferty (2010). "High-Dimensional Ising Model Selection Using *L*_1-Regularized Logistic Regression". In: *Annals of Statistics* 38(3), pages 1287–1319 (cited on pp. 15, 22, 27, 37).

Matthew Richardson and Pedro Domingos (2006). "Markov Logic Networks". In: *Machine Learning* 62(1–2), pages 107–136 (cited on pp. 16, 17, 45).

David Silver, Aja Huang, Chris Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis (2016). "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* 529(7587), pages 484–489 (cited on p. 3).

Parag Singla and Pedro Domingos (2005). "Discriminative Training of Markov Logic Networks". In: *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005; Pittsburgh, Pennsylvania, United States; 9-13 July 2005)*, pages 868–873 (cited on p. 17).

Ashwin Srinivasan (2001). "The Aleph Manual". In: *Machine Learning at the Computing Laboratory, Oxford University* (cited on pp. 11, 83, 98).

Alexander van den Berghe, Jan Van Haaren, Stefan Van Baelen, Yolande Berbers, and Wouter Joosen (2013). "Towards an Automated Pattern Selection Procedure in Software Models". In: *Late Breaking Papers of the Twenty-Second International Conference on Inductive Logic Programming (ILP 2012; Dubrovnik, Croatia; 17-19 September 2012)*, pages 68–73 (cited on p. 128).

Guy Van den Broeck (2011). "On the Completeness of First-Order Knowledge Compilation for Lifted Probabilistic Inference". In: *Advances in Neural Information Processing Systems 24 (NIPS 2011; Granadam Spain; 12-17 December 2011)*, pages 1386–1394 (cited on p. 17).

Guy Van den Broeck and Adnan Darwiche (2013). "On the Complexity and Approximation of Binary Evidence in Lifted Inference". In: *Advances in Neural Information Processing Systems 26 (NIPS 2013; South Lake Tahoe, Nevada, United States; 5-10 December 2013)*, pages 2868–2876 (cited on p. 46).

Guy Van den Broeck, Wannes Meert, and Jesse Davis (2013). "Lifted Generative Parameter Learning". In: *Proceedings of the Third International Workshop on Statistical Relational AI (StaRAI 2013; Bellevue, Washington, United States; 15 July 2013)* (cited on pp. 17, 40, 41, 44, 47).

Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt (2011). "Lifted Probabilistic Inference by First-Order Knowledge Compilation". In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011; Barcelona, Spain; 16-22 July 2011)*, pages 2178–2185 (cited on p. 47).

Jan Van Haaren (2012). "Analyzing Football Matches Using Relational Performance Data". In: *Workshop on Mining Football-related Data (MFD 2012; Amsterdam, The Netherlands; 7 June 2012)* (cited on p. 130).

Jan Van Haaren (2014). *Waarom onze Rode Duivels maar best snel twee keer scoren tegen Algerije*. URL: https://www.kuleuvenblogt.be/2014/06/16/waarom-onze-rode-duivels-maar-best-snel-twee-keer-scoren-tegen-algerije (cited on p. 130).

Jan Van Haaren (2015). *Statistische simulatie van de Belgische Pro League 2014-2015: De samenstelling van de play-offs voorspeld*. CW Reports CW682. Leuven, Belgium: Department of Computer Science, KU Leuven (cited on p. 129).

Jan Van Haaren (2016). *België heeft 7,6% kans op EK-titel*. URL: https://www.kuleuvenblogt.be/2016/06/10/belgie-heeft-76-kans-op-ek-titel (cited on p. 130).

Jan Van Haaren, Horesh Ben Shitrit, Jesse Davis, and Pascal Fua (2016). "Analyzing Volleyball Match Data from the 2014 World Championships Using Machine Learning Techniques". In: *Proceedings of the Twenty-Second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016; San Francisco, California, United States; 13-17 August 2016)* (cited on pp. 7, 92, 128).

Jan Van Haaren and Jesse Davis (2012). "Markov Network Structure Learning: A Randomized Feature Generation Approach". In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012; Toronto, Ontario, Canada; 22-26 July 2012)*, pages 1148–1154 (cited on pp. 6, 22, 127).

Jan Van Haaren and Jesse Davis (2014). *Performance Analysis of the 2014 FIFA World Cup Group Stage*. CW Reports CW673. Leuven, Belgium: Department of Computer Science, KU Leuven (cited on p. 129).

Jan Van Haaren and Jesse Davis (2015a). "Predicting the Final League Tables of Domestic Football Leagues". In: *Proceedings of the Fifth International Conference on Mathematics in Sport (MathSport International; Loughborough, United Kingdom; 29 June - 1 July 2015)*, pages 202–207 (cited on p. 128).

Jan Van Haaren and Jesse Davis (2015b). *Prestatie-analyse van de clubs in de Belgische Pro League 2014-2015: De reguliere competitie doorgelicht*. CW Reports CW683. Leuven, Belgium: Department of Computer Science, KU Leuven (cited on p. 129).

Jan Van Haaren, Jesse Davis, Martijn Lappenschaar, and Arjen Hommersom (2013). "Exploring Disease Interactions Using Markov Networks". In: *Proceedings of the AAAI 2013 Workshop on Expanding the Boundaries of Health Informatics Using AI (HIAI 2013; Bellevue, Washington, United States; 15 July 2013)* (cited on p. 129).

Jan Van Haaren, Vladimir Dzyuba, Siebe Hannosset, and Jesse Davis (2015). "Automatically Discovering Offensive Patterns in Soccer Match Data". In: *Advances in Intelligent Data Analysis XIV (IDA 2015; Saint-Étienne, France; 22-24 October 2015)*, pages 286–297 (cited on pp. 7, 78, 128).

Jan Van Haaren, Siebe Hannosset, and Jesse Davis (2016). "Strategy Discovery in Professional Soccer Match Data". In: *Proceedings of the KDD 2016 Workshop on Large-Scale Sports Analytics (LSSA 2016; San Francisco, California, United States; 14 August 2016)* (cited on p. 129).

Jan Van Haaren, Andrey Kolobov, and Jesse Davis (2015). "TODTLER: Two-Order-Deep Transfer Learning". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015; Austin, Texas, United States; 25-30 January 2015)*, pages 3007–3015 (cited on pp. 7, 56, 127).

Jan Van Haaren, Tim Op De Beéck, and Jesse Davis (2014a). *Prestatie-analyse van de clubs in de Belgische Pro League 2013-2014: De play-offs doorgelicht*. CW Reports CW665. Leuven, Belgium: Department of Computer Science, KU Leuven (cited on p. 129).

Jan Van Haaren, Tim Op De Beéck, and Jesse Davis (2014b). *Prestatie-analyse van de clubs in de Belgische Pro League 2013-2014: De reguliere competitie doorgelicht*. CW Reports CW658. Leuven, Belgium: Department of Computer Science, KU Leuven (cited on p. 129).

Jan Van Haaren and Guy Van den Broeck (2011). "Relational Learning for Football-related Predictions". In: *Latest Advances in Inductive Logic Programming (ILP 2011; Windsor Great Park, United Kingdom; 31 July - 3 August 2011)*, pages 237–244 (cited on p. 128).

Jan Van Haaren and Guy Van den Broeck (2012). "Relational Learning for Football-related Predictions". In: *Proceedings of the Twenty-First Belgian-Dutch*

Conference on Machine Learning (BeNeLearn 2012; Ghent, Belgium; 24-25 May 2012), page 85 (cited on p. 130).

Jan Van Haaren, Guy Van den Broeck, Wannes Meert, and Jesse Davis (2014c). "Tractable Learning of Liftable Markov Logic Networks". In: *Proceedings of the ICML 2014 Workshop on Learning Tractable Probabilistic Models (LTPM 2014; Beijing, China; 26 June 2014)* (cited on pp. 7, 41, 129).

Jan Van Haaren, Guy Van den Broeck, Wannes Meert, and Jesse Davis (2016). "Lifted Generative Learning of Markov Logic Networks". In: *Machine Learning* 103(1), pages 27–55 (cited on pp. 7, 41, 127).

Jan Van Haaren, Albrecht Zimmermann, Joris Renkens, Guy Van den Broeck, Tim Op De Beéck, Wannes Meert, and Jesse Davis (2013). "Machine Learning and Data Mining for Sports Analytics". In: *LStat 25th Anniversary Scientific Event (LStat 25; Leuven, Belgium; 13-14 December 2013)* (cited on p. 130).

Stefan Wrobel (1997). "An Algorithm for Multi-Relational Discovery of Subgroups". In: *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD 1997; Trondheim, Norway; 24-27 June 1997)*, pages 78–87 (cited on p. 79).

Curriculum Vitae

Jan Van Haaren was born on March 11th 1988 in Turnhout and grew up in Weelde. He attended high school at the Sint-Pietersinstituut in Turnhout. He obtained a Bachelor of Science in Informatics degree from KU Leuven in 2010 and a Master of Science in Engineering degree specialized in computer science and artificial intelligence from the same university in 2011. His Master's thesis was entitled *Relational Learning for Football-related Predictions*.

He began doctoral studies under the supervision of Prof. dr. Jesse Davis in the Machine Learning group within the Declarative Languages and Artificial Intelligence lab at KU Leuven in September 2011. He received a four-year PhD fellowship from the Agency for Innovation by Science and Technology in Flanders (IWT) in 2013. He was a visiting student for three months in the Computer Vision lab of Prof. dr. Pascal Fua at the École Polytechnique Fédérale de Lausanne (EPFL) in 2015. He will defend his doctoral dissertation entitled *Relational Approaches for Learning, Transferring and Mining* in December 2016.

List of Publications

Journal Article

Jan Van Haaren, Guy Van den Broeck, Wannes Meert, and Jesse Davis (2016). "Lifted Generative Learning of Markov Logic Networks". In: *Machine Learning* 103(1), pages 27–55

Highly-Selective Conference Papers

Jan Van Haaren and Jesse Davis (2012). "Markov Network Structure Learning: A Randomized Feature Generation Approach". In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012; Toronto, Ontario, Canada; 22-26 July 2012)*, pages 1148–1154

Jan Van Haaren, Andrey Kolobov, and Jesse Davis (2015). "TODTLER: Two-Order-Deep Transfer Learning". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015; Austin, Texas, United States; 25-30 January 2015)*, pages 3007–3015

Jan Van Haaren, Horesh Ben Shitrit, Jesse Davis, and Pascal Fua (2016). "Analyzing Volleyball Match Data from the 2014 World Championships Using Machine Learning Techniques". In: *Proceedings of the Twenty-Second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016; San Francisco, California, United States; 13-17 August 2016)*

Other Conference Papers

Jan Van Haaren, Vladimir Dzyuba, Siebe Hannosset, and Jesse Davis (2015). "Automatically Discovering Offensive Patterns in Soccer Match Data". In: *Advances in Intelligent Data Analysis XIV (IDA 2015; Saint-Étienne, France; 22-24 October 2015)*, pages 286–297

Jan Van Haaren and Jesse Davis (2015a). "Predicting the Final League Tables of Domestic Football Leagues". In: *Proceedings of the Fifth International Conference on Mathematics in Sport (MathSport International; Loughborough, United Kingdom; 29 June - 1 July 2015)*, pages 202–207

Jan Van Haaren and Guy Van den Broeck (2011). "Relational Learning for Football-related Predictions". In: *Latest Advances in Inductive Logic Programming (ILP 2011; Windsor Great Park, United Kingdom; 31 July - 3 August 2011)*, pages 237–244

Alexander van den Berghe, Jan Van Haaren, Stefan Van Baelen, Yolande Berbers, and Wouter Joosen (2013). "Towards an Automated Pattern Selection Procedure in Software Models". In: *Late Breaking Papers of the Twenty-Second International Conference on Inductive Logic Programming (ILP 2012; Dubrovnik, Croatia; 17-19 September 2012)*, pages 68–73

Tim Op De Beéck, Arjen Hommersom, Jan Van Haaren, Maarten van der Heijden, Jesse Davis, Peter Lucas, Lucy Overbeek, and Iris Nagtegaal (2015). "Mining Hierarchical Pathology Data Using Inductive Logic Programming". In: *Proceedings of the Fifteenth Conference on Artificial Intelligence in Medicine (AIME 2015; Pavia, Italy; 17-20 June 2015)*, pages 76–85

Workshop Papers

Jan Van Haaren, Guy Van den Broeck, Wannes Meert, and Jesse Davis (2014c). "Tractable Learning of Liftable Markov Logic Networks". In: *Proceedings of the ICML 2014 Workshop on Learning Tractable Probabilistic Models (LTPM 2014; Beijing, China; 26 June 2014)*

Jan Van Haaren, Siebe Hannosset, and Jesse Davis (2016). "Strategy Discovery in Professional Soccer Match Data". In: *Proceedings of the KDD 2016 Workshop on Large-Scale Sports Analytics (LSSA 2016; San Francisco, California, United States; 14 August 2016)*

Jan Van Haaren, Jesse Davis, Martijn Lappenschaar, and Arjen Hommersom (2013). "Exploring Disease Interactions Using Markov Networks". In: *Proceedings of the AAAI 2013 Workshop on Expanding the Boundaries of Health Informatics Using AI (HIAI 2013; Bellevue, Washington, United States; 15 July 2013)*

Technical Reports

Jan Van Haaren and Jesse Davis (2014). *Performance Analysis of the 2014 FIFA World Cup Group Stage*. CW Reports CW673. Leuven, Belgium: Department of Computer Science, KU Leuven

Jan Van Haaren, Tim Op De Beéck, and Jesse Davis (2014b). *Prestatie-analyse van de clubs in de Belgische Pro League 2013-2014: De reguliere competitie doorgelicht*. CW Reports CW658. Leuven, Belgium: Department of Computer Science, KU Leuven

Jan Van Haaren, Tim Op De Beéck, and Jesse Davis (2014a). *Prestatie-analyse van de clubs in de Belgische Pro League 2013-2014: De play-offs doorgelicht*. CW Reports CW665. Leuven, Belgium: Department of Computer Science, KU Leuven

Jan Van Haaren and Jesse Davis (2015b). *Prestatie-analyse van de clubs in de Belgische Pro League 2014-2015: De reguliere competitie doorgelicht*. CW Reports CW683. Leuven, Belgium: Department of Computer Science, KU Leuven

Jan Van Haaren (2015). *Statistische simulatie van de Belgische Pro League 2014-2015: De samenstelling van de play-offs voorspeld*. CW Reports CW682. Leuven, Belgium: Department of Computer Science, KU Leuven

Abstracts

Jan Van Haaren and Guy Van den Broeck (2012). "Relational Learning for Football-related Predictions". In: *Proceedings of the Twenty-First Belgian-Dutch Conference on Machine Learning (BeNeLearn 2012; Ghent, Belgium; 24-25 May 2012)*, page 85

Jan Van Haaren (2012). "Analyzing Football Matches Using Relational Performance Data". In: *Workshop on Mining Football-related Data (MFD 2012; Amsterdam, The Netherlands; 7 June 2012)*

Jan Van Haaren, Albrecht Zimmermann, Joris Renkens, Guy Van den Broeck, Tim Op De Beéck, Wannes Meert, and Jesse Davis (2013). "Machine Learning and Data Mining for Sports Analytics". In: *LStat 25th Anniversary Scientific Event (LStat 25; Leuven, Belgium; 13-14 December 2013)*

Popularizing Articles

Jan Van Haaren (2014). *Waarom onze Rode Duivels maar best snel twee keer scoren tegen Algerije*. URL: https://www.kuleuvenblogt.be/2014/06/16/waarom-onze-rode-duivels-maar-best-snel-twee-keer-scoren-tegen-algerije

Jan Van Haaren (2016). *België heeft 7,6% kans op EK-titel*. URL: https://www.kuleuvenblogt.be/2016/06/10/belgie-heeft-76-kans-op-ek-titel

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
DECLARATIVE LANGUAGES AND ARTIFICIAL INTELLIGENCE
Celestijnenlaan 200A box 2402
B-3001 Leuven
jan.vanhaaren@cs.kuleuven.be
http://people.cs.kuleuven.be/~jan.vanhaaren