Deep Transfer as Structure Learning in Markov Logic Networks

David Moore and **Andrea Danyluk**
Williams College
Williamstown, MA
{10dam, andrea}@cs.williams.edu

Abstract

Learning the relational structure of a domain is a fundamental problem in statistical relational learning. The deep transfer algorithm of Davis and Domingos attempts to improve structure learning in Markov logic networks by harnessing the power of transfer learning, using the second-order structural regularities of a source domain to bias the structure search process in a target domain. We propose that the clique-scoring process which discovers these second-order regularities constitutes a novel standalone method for learning the structure of Markov logic networks, and that this fact, rather than the transfer of structural knowledge across domains, accounts for much of the performance benefit observed via the deep transfer process. This claim is supported by experiments in which we find that clique scoring within a single domain often produces results equaling or surpassing the performance of deep transfer incorporating external knowledge, and also by explicit algorithmic similarities between deep transfer and other structure learning techniques.

Introduction

The growing field of statistical relational learning aims to develop methods for learning, inference, and dealing with uncertainty in domains which are fundamentally relational, in contrast to traditional statistical learning techniques which are often limited to propositional data. Markov logic networks (MLNs) are a representation which generalizes first-order logic and probabilistic graphical models, using weighted formulas of first-order logic to represent knowledge about a relational domain (Richardson and Domingos 2006). Learning an MLN representation for a domain can be reduced to two problems: *structure learning*, i.e. discovering a set of logical formulas, and *weight learning*, i.e. choosing a weight for each formula. Learning the structure of an MLN for a relational domain is an important problem for which several algorithms have been proposed (for example, Kok and Domingos 2005; Mihalkova and Mooney 2007; Biba, Ferilli, and Esposito 2008; Kok and Domingos 2009).

One approach to improving structure learning performance is to apply *transfer learning*, which uses knowledge gained from learning a source task to aid learning in a related target task (Torrey and Shavlik 2009). The deep transfer algorithm (DTM) proposed by Davis and Domingos transfers knowledge between relational domains, which may be composed of entirely different predicates, by analyzing structural tendencies of formulas in the source domain and representing them as domain-independent knowledge in the form of second-order cliques. The cliques are scored to identify the most salient structural tendencies, and the highest-scoring cliques are instantiated as first-order formulas in the target domain, where they provide a declarative bias for structure learning. DTM has been shown to deliver improved learning performance for several pairings of source and target domains, relative to pure structure learning within the target domain.

In this paper, we argue that the performance increases observed through DTM are not due primarily to its role in transferring knowledge across domains, but instead that the the algorithm is in fact performing a novel form of structure learning, which we will refer to as learning through clique scoring with greedy selection (CSGL). As evidence, we show that the DTM algorithm can be applied within a single domain to obtain performance equalling or surpassing the best cases of transfer, but without incorporating external knowledge. Furthermore, although DTM (and thus CSGL) can make use of the MSL structure learning algorithm (Kok and Domingos 2005) to refine learned theories, we show that even when this step is omitted, structure learning through CSGL produces results equalling or exceeding those of MSL.

We begin by briefly reviewing Markov logic and the deep transfer algorithm. We then describe the role of clique scoring in deep transfer and explain how, in conjunction with first-order instantiation and greedy selection, it can be construed as an algorithm for structure learning. We demonstrate empirically that clique scoring can be used effectively to learn in a target domain even without a distinct source domain, and we evaluate its performance as a standalone structure learning algorithm. Finally, we discuss the relationship between CSGL and other structure learning algorithms, and note in particular its strong parallels with a version of the hypergraph lifting approach (LHL) presented in (Kok and

Domingos 2009).

Markov Logic and Deep Transfer

Terminology

In first-order logic, a constant refers to a particular object in the domain. Variables stand in place of constants for the purpose of quantification. A term is a constant or a variable. A predicate represents a relation on objects in the domain and is either true or false for any given list of objects; the arity of a predicate is the number of arguments it takes. An atom is a predicate applied to a list of terms, and a ground atom is an atom in which all of the terms are constants. A clause is a disjunction of atoms, any of which may be negated; a formula may also include conjunction and implication. A ground formula is a formula in which all of the atoms are ground atoms. A world (or "possible world") is an assignment of truth values to all ground atoms in the domain.

Markov Logic Networks

Markov logic networks (MLNs), introduced by Richardson and Domingos (2006), are a knowledge representation generalizing both first-order logic and probabilistic graphical models. An MLN is a set of first-order formulas with associated weights. Given a finite set of constants, an MLN may be represented as a Markov network (Pearl 1988) consisting of a node for each ground atom in the domain and an edge between any two nodes representing ground atoms which appear together in the same ground formula. In this representation, the cliques of the graph correspond roughly to the ground formulas of the domain, and to each ground formula j we assign a feature $g_j(x)$, which is defined to be 1 if formula j is true in world x and 0 otherwise. We also assign a weight w_j equal to the weight of the corresponding first-order formula in the MLN. Markov networks model the joint probability distribution of a set of variables, and so the Markov network created from a grounded MLN can be interpreted as defining a probability distribution over the set of all possible worlds X, given by $P(X = x) = \frac{1}{Z} \exp\left(\sum_j w_j g_j(x)\right)$, where the sum is taken over all features in the graph and Z is a normalization constant. This formula can be written equivalently as

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{i \in F} w_i n_i(x)\right),$$

where F is the set of all first-order formulas in the MLN, w_i is the weight of the ith formula, and n_i is the number of true groundings of the ith formula in possible world x; note that this rendering is in terms of the first-order MLN structure rather than the underlying Markov network structure. By generalizing the propositional representation of Markov networks to include first-order logic, MLNs are capable of succinctly and intuitively representing complex models involving uncertainty in relational domains.

Several algorithms have been proposed for structure learning in MLNs. MSL, proposed by Kok and Domingos (2005), uses an inductive logic programming-style beam search in which atoms to be added or removed from a clause are evaluated for their effect on the weighted pseudo-log-likelihood (WPLL, defined by Kok and Domingos 2005) of the data. As a local search strategy, MSL has the useful property that it can be used both to learn structure from scratch and also to refine existing theories. More recently, several algorithms have been proposed which may outperform MSL in learning performance, required training time, or both; see Mihalkova and Mooney (2007), Biba, Ferilli, and Esposito (2008), and Kok and Domingos (2009).

Deep Transfer

The goal of deep transfer learning, as described by Davis and Domingos (2009), is to identify domain-independent abstract knowledge which can be used to bias the process of structure learning in a destination domain. The process begins with an MLN representing a source domain, abstracts the MLN into second-order cliques which represent second-order structure, identifies the cliques which correspond to the most salient structural properties, and then uses those to bias structure learning of an MLN in the target domain. The initial source-domain MLN can be generated by any method, but Davis and Domingos obtained their best results when using exhaustive search, which considers all possible clauses under a maximum length and number of object variables.

Second-order cliques are simply sets of second-order predicates which occur together in at least one clause (e.g. the first-order clauses `!Friends(x,y) ∨ Friends(y,x)` and `!Enemies(x,y) ∨ Enemies(y,x)` each correspond to the same second-order clique {`r(x,y)`, `r(y,x)`}. To score a clique, the algorithm examines each of its first-order instantiations. Each instantiation is decomposed into all possible pairs of subcliques, and each pair is assigned a score equal to the K-L divergence $D(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$, where p is the probability distribution of the entire clique (for any state of the clique x, $p(x)$ gives the probability that the clique will be in state x if we instantiate its terms with constants from the domain) and q is the distribution it would have if the two subcliques were independent. The score of a first-order instantiation is the minimum score of all of its decompositions, and the score of the second-order clique is the average of its top n instantiations, which encourages cliques having multiple useful instantiations.

Once the scoring process is complete, DTM selects the top k second-order cliques which have at least one true grounding in the target domain, and instantiates them by creating all corresponding first-order clauses in the target domain. These clauses include all possible ways of flipping the signs of the atoms within the clique. In the "greedy transfer with refinement" approach, which Davis and Domingos found to be the most effective, DTM creates an MLN from the resulting clauses by first greedily selecting clauses to improve the WPLL until no additional clause does so, and then applying MSL to refine the theory suggested by the greedy procedure.

DTM for Structure Learning

Although DTM can extract cliques from any set of formulas in the source domain, Davis and Domingos found that the best results came from considering not the results of beam search or other structure learning techniques, but a simple exhaustive listing of all possible clauses in the domain within a maximum length and number of terms. They argue that since the clique scoring process already suggests the most useful cliques for transfer, there is no additional benefit in trying to learn a theory in the source domain which would restrict the cliques that are considered for transfer. In this paper we will generally use DTM to refer to the form of the algorithm in which exhaustive search is used to generate cliques for transfer.

An interesting effect of the use of exhaustive search is that unlike many transfer methods, DTM performs a different sort of learning in the source domain than in the target domain. DTM's only interaction with the data of the source domain is through the clique-scoring process, while in the destination domain it uses a greedy selection process, followed optionally by beam search, to refine the clauses produced through transfer. This raises the question of whether the demonstrated benefits of transfer can be attributed entirely to the use of source domain knowledge, or whether the clique scoring process might be acting as a novel form of structure learning in itself, providing a performance boost independent of the particular structural knowledge being imported from the source domain.

We propose a simple modification to DTM, which we call "self-transfer". The inspiration for self-transfer is related to the persistent question in transfer learning of how to know when two tasks are sufficiently related that transfer between them will be worthwhile. For transfer to be at its most effective, we generally want the source and target tasks to be as similar as possible. Since no two domains are more similar to each other than a domain is to itself, we propose a method that effectively applies DTM to a single domain as both source and target. We first construct an exhaustive list of all possible first-order clauses in the domain up to some maximum length and number of object variables. These clauses are then abstracted into second-order cliques and scored via the method described above. The k top-scoring cliques are then instantiated back into first-order clauses of the domain. From this set of clauses, we use the "greedy transfer with refinement" method described by Davis and Domingos to derive a final set of clauses indicating the structure of the domain. That is, we greedily select from the instantiated first-order clauses until no additional clause improves the WPLL, and then we refine the domain theory with MSL.

One might expect that using the same domain as both source and target would forfeit some of the expected benefits of transfer, for example the improved generalization performance which often results from considering a broader class of related tasks (Caruana 1997). Surprisingly, this does not appear to be the case: in our experiments the results of self-transfer were generally comparable to the best results from DTM. While there may exist circumstances in which cross-domain transfer through DTM is more effective (e.g. if the target domain has very little data available, or if we wish to learn structure for many domains using the same source domain to avoid repeating the clique-scoring process for each domain), we demonstrate that in many cases the benefits of transfer observed using DTM are achievable through simple self-transfer.

This observation strongly suggests that much of the utility of DTM comes, not from the ability to transfer knowledge between domains, but from the introduction of a new learning process, namely the process of clique scoring in conjunction with first-order instantiation and greedy selection. In fact, we find that clique scoring with greedy selection (CSGL) alone yields results equalling or exceeding those of MSL.

	Types	Predicates	Constants	True Atoms	Total Atoms
IMDB	5	5	251	1039	55722
UW-CSE	8	10	442	1407	174785
WebKB	3	3	4953	283489	20663916
Yeast Protein	7	7	2470	15015	4533900

Table 1: Datasets used.

Experiments

To evaluate the effectiveness of self-transfer relative to other transfer cases, as well as the effectiveness of CSGL as a standalone structure learning algorithm, we performed experiments using data from several real-world domains. We used the implementation of MSL in the publicly available Alchemy package (Kok et al. 2009) along with an Alchemy-based implementation of DTM provided by Jesse Davis.

Domains

For comparison with the results of Davis and Domingos (2009), we used datasets representing the WebKB and Yeast Protein domains. Both datasets were provided by Jesse Davis.

WebKB. This dataset consists of labeled web pages from the computer science departments of four universities, with predicates indicating the words occurring on each page, the class label of each page (faculty, student, course, etc.), and the links between pages. The data from each university is treated as a separate fold. We attempt to predict the truth values of all groundings of the `Linked` and `PageClass` predicates. The data is originally sourced from Craven and Slattery (2001), and the version used both in this paper and in (Davis and Domingos 2009) is equivalent to the version publicly available at `alchemy.cs.washington.edu`, with the following modifications: the seven single-arity predicates `FacultyPage`, `CoursePage`, etc. indicating the class label of a page are collapsed into a single predicate `PageClass` of arity two, and only the `Has`, `Linked`, and `PageClass` predicates are considered.

Yeast Protein. This dataset contains information on protein location, function, phenotype, class, and enzymes within the yeast *Saccharomyces cerevisiae*, as well as protein interactions and protein complex data, all from the

MIPS Comprehensive Yeast Genome Database as of February 2005 (Mewes et al. 2002). We used the version of the data from Davis and Domingos, which is split into four disjoint subsamples which are used as folds, and we attempt to predict the `Interaction` and `Function` predicates.

We also used two additional domains, both publicly available from `alchemy.cs.washington.edu`:

IMDB. This dataset describes a movie domain, consisting of movies, actors, directors, etc. and predicates indicating their relationships. The data was collected from `imdb.com` by Mihalkova and Mooney (2007). The data is split into five disjoint folds, but in order to maintain consistency with WebKB and Yeast, we used only the first four folds. We attempt to predict the `WorkedUnder` and `WorkedInGenre` predicates. Following Kok and Domingos (2009) we omitted four equality predicates which are superseded by the equality operator available in Alchemy. In addition, for consistency with WebKB, we collapsed the single-arity `Actor` and `Director` predicates into a single predicate `HasRole` with arity two (similar to `PageClass` in WebKB).

UW-CSE. This dataset, from Richardson and Domingos (2006), describes anonymized relationships between students, faculty, and courses in the University of Washington Computer Science and Engineering Department. The data is split into five folds representing the sub-disciplines of AI, graphics, programming languages, systems, and theory; again for consistency we used only four folds, omitting the systems data (chosen randomly). Following Richardson and Domingos we attempted to predict the `AdvisedBy` predicate. As with IMDB, we omitted nine redundant equality predicates, and we collapsed the single-arity `Student` and `Professor` into a single `HasRank`. We also simplified the `TaughtBy` and `TA` predicates of UW-CSE to ignore the particular quarter in which a course was taught, reducing the arity of each of those predicates from three to two. This change was motivated by the fact that DTM can only transfer between predicates having the same arity; because there are no arity-three predicates in our other datasets these predicates would otherwise have been completely ignored by the transfer process.

Details of all datasets are given in Table 1.

Experiment 1: DTM vs. Self-Transfer

To compare the effectiveness of cross-domain transfer to that of self-transfer, we used DTM to perform transfer between all combinations of source and target domains, including cases of self-transfer. Each dataset was divided into four independent folds, on which we performed leave-one-out cross-validation, training on every subset of three folds and testing on the fourth. The results represent averages over the four folds from each domain. For tractability on the WebKB data, we followed Davis and Domingos in using information gain on the training set to pick the fifty words most predictive of page class; these were used to train and evaluate the learned MLN.

In cases of self-transfer, we gathered and scored cliques using only the training set, not the full dataset. Since in all other transfer cases we gathered and scored cliques using all data available from the source domain, this puts self-transfer at a modest disadvantage in terms of the quantity of data available during the learning process. That said, within each domain the cliques which were identified in the three folds of each training set did not differ significantly from those obtained using the full four folds, with only minor changes in ordering in most cases.

Following Davis and Domingos, we allowed MSL to learn clauses containing constants. We permitted only role (IMDB), rank (UW-CSE), function (Yeast), and page class (WebKB) to appear as constants in learned clauses. We evaluated DTM with $k = 5$ and $k = 10$; because the results show the same overall trends we report only the $k = 10$ results here. Source clauses for DTM were generated by exhaustive search over all clauses containing at most three literals and three object variables. MSL structure refinement was time-limited to 20 hours for each trial. Following Kok and Domingos (2005) and others, we evaluated each set of learned formulas using the test set conditional log-likelihood (CLL) and the area under the precision-recall curve (AUC). The CLL has the advantage of directly measuring the quality of the probability estimates produced, while the AUC is useful because it is insensitive to the large number of true negatives in the data. The CLL is calculated by averaging over all ground atoms the predicted log-likelihood that each ground atom takes on its true value, given the learned domain theory and the truth values of all other ground atoms as evidence. AUC is calculated by varying the threshold CLL above which an atom is predicted to be true.

Experiment 2: CSGL vs. MSL

We measured the performance of clique scoring with greedy selection as a standalone structure learning algorithm by comparing its performance to that of MSL. Results for CSGL on each domain were obtained by omitting the refinement step from the self-transfer results of the previous section, that is, by combining exhaustive search with clique scoring followed by greedy selection. Evaluation was performed identically to the previous section; we report results for $k = 5$ and $k = 10$.

Results

Table 2 gives the AUC and CLL for all transfer scenarios using refinement, including self-transfer results (which are underlined) as well as MSL, which acts as a baseline. Each figure represents an average over the four different train/test trials. Note that the results for transfer from WebKB and Yeast are identical. This is because the ten highest-scoring cliques are the same in both domains (see Figure 4 for a listing of the top cliques in each domain), so deep transfer produces the same theories when transferring from either domain. Also note that the top five cliques are also identical across WebKB and Yeast.

The results in Table 2 support the claims of Davis and Domingos, in that DTM improves on MSL in almost all cases. Examining self-transfer in particular, we see that self-transfer is at least competitive with other transfer scenarios on all predicates except `PageClass` (where the deficiency is due to a single outlier trial, in which refinement of the results from greedy selection led to a large decrease

	IMDB	UW-CSE	WebKB	Yeast	MSL
WorkedInGenre	0.63	0.61	0.61	0.61	0.32
WorkedUnder	0.77	0.23	0.21	0.21	0.03
AdvisedBy	0.08	0.08	0.08	0.08	0.04
Linked	0.01	0.01	0.09	0.09	0.004
PageClass	0.86	0.86	0.68	0.68	0.87
Function	0.34	0.34	0.33	0.33	0.27
Interaction	0.04	0.04	0.10	0.10	0.04

(a) AUC

	IMDB	UW-CSE	WebKB	Yeast	MSL
WorkedInGenre	-0.20	-0.13	-0.37	-0.37	-0.30
WorkedUnder	-0.09	-0.17	-0.21	-0.21	-0.23
AdvisedBy	-0.03	-0.03	-0.03	-0.03	-0.04
Linked	-0.02	-0.02	-0.02	-0.02	-0.02
PageClass	-0.07	-0.07	-0.12	-0.12	-0.07
Function	-0.18	-0.18	-0.18	-0.18	-0.19
Interaction	-0.04	-0.04	-0.03	-0.03	-0.04

(b) CLL

Table 2: Results for DTM vs. self-transfer (underlined).

	CSGL-5	CSGL-10	MSL
WorkedInGenre	0.70	0.63	0.32
WorkedUnder	0.26	0.69	0.03
AdvisedBy	0.04	0.06	0.04
Linked	0.06	0.06	0.004
PageClass	0.86	0.86	0.87
Function	0.31	0.31	0.27
Interaction	0.10	0.10	0.04

(a) AUC

	CSGL-5	CSGL-10	MSL
WorkedInGenre	-0.16	-0.15	-0.30
WorkedUnder	-0.14	-0.11	-0.23
AdvisedBy	-0.04	-0.03	-0.04
Linked	-0.02	-0.02	-0.02
PageClass	-0.07	-0.07	-0.07
Function	-0.17	-0.17	-0.19
Interaction	-0.03	-0.03	-0.04

(b) CLL

Table 3: Results for CSGL (top 5 and top 10 cliques) vs. MSL.

in AUC), and that in fact it performs significantly better than all other methods in AUC when predicting the predicate `WorkedUnder` (paired one-tail t-test, $p < 0.05$). For no predicate does the best case of cross-domain transfer perform significantly better, in AUC or CLL, than self-transfer does (paired one-tail t-test, $p > 0.10$). This is consistent with our claim that the performance gains of DTM over MSL are not related to DTM's incorporation of source-domain knowledge.

Note that self-transfer generally matches or outperforms other transfer settings despite the limitation of having only the three folds of the training set from which to identify the top cliques, as opposed to using the full four folds of source domain data which are available to the other transfer scenarios. Also recall that we modified the logical structure of each dataset so that all of its predicates had arity two, thus giving DTM the greatest possible freedom to transfer structure between all predicates. If we had allowed the single-arity `Student` and `Professor` predicates to remain uncollapsed in the UW-CSE dataset, for example, then DTM would have been unable to relate them to the analogous `PageClass` predicate in WebKB because it has arity two. By contrast, self-transfer generates cliques with arities appropriate to the predicates of each dataset.

Table 3 compares CSGL to MSL as standalone structure learning algorithms, with two versions of CSGL instantiating the top five and ten highest-scoring cliques respectively. Results from CSGL-5 and CSGL-10 were generally comparable, although CSGL-10 fared much better when predicting `WorkedUnder`. Note that CSGL-10 beats MSL in every case except for the `PageClass` predicate of WebKB, for which the two methods give approximately equal results. CGSL-10 also performs comparably to 10-clique self-transfer in most cases, and substantially better in the case `PageClass`, indicating that the additional, costly refinement step required by self-transfer may not be necessary in order to achieve satisfactory results.

Related Work

Several structure learning algorithms have been proposed for Markov logic networks, but of particular relevance here is LHL, the hypergraph lifting approach described by Kok and Domingos (2009). This is because the unlifted variant of LHL, known as LHL-FindPaths, bears remarkable similarities to CSGL in its general structure. Like LHL-FindPaths, CSGL is a bottom-up structure learner which constructs clauses directly from the data. The exhaustive search step used by CSGL to generate initial clauses is equivalent to the process in LHL-FindPaths of enumerating and variabilizing paths in the unlifted hypergraph, except for the added restriction that every conjunction which LHL-FindPaths considers must have at least one support in the data. Both methods evaluate clauses according to how well they represent structural regularities not found in their sub-clauses; in CSGL this is implemented by the clique-scoring process in which all but the top-scoring cliques are discarded, while in LHL this is done by simply discarding any clause having a WPLL less than one of its subclauses. Both methods consider as candidates many combinations of negated and non-negated atoms in the clauses that they generate; in CSGL this is part of the clique abstraction and instantiation process, while LHL explicitly constructs partially-negated variants of its clauses. Finally, both methods arrive at the final MLN structure by greedily selecting clauses from a list of candidates until no

Rank	IMDB	UW-CSE	WebKB	Yeast
1	r(x,y),r(x,z)	r(x,y),r(x,z)	r(x,y),r(z,y)	r(x,y),r(z,y)
2	r(x,y),r(z,y)	r(x,y),r(z,y)	r(x,y),r(x,z)	r(x,y),r(x,z)
3	r(x,y),r(z,y),s(x,z)	r(x,y),r(z,y),s(x,z)	r(x,y),r(z,y),s(x,x)	r(x,y),r(y,x)
4	r(x,y),r(y,z),r(z,x)	r(x,y),r(z,y),s(x,x)	r(x,y),r(z,y),s(x,z)	r(x,y),r(z,y),s(x,z)
5	r(x,y),r(x,z),s(x,x)	r(x,y),r(x,z),s(x,x)	r(x,y),r(y,x)	r(x,y),r(z,y),s(x,x)
6	r(x,y),r(z,y),s(x,x)	r(x,x),s(x,x)	r(x,y),r(y,z),r(z,x)	r(x,y),r(y,z),r(z,x)
7	r(x,y),s(y,z)	r(x,y),s(x,z)	r(x,x),r(x,y),r(y,x)	r(x,y),r(x,z),r(y,x)
8	r(x,y),r(x,z),s(y,z)	r(x,x),s(x,x),t(x,y)	r(x,y),r(y,x),s(x,z)	r(x,y),r(y,x),s(x,z)
9	r(x,y),s(x,z)	r(x,y),s(y,z)	r(x,y),r(x,z),r(y,x)	r(x,x),r(x,y),r(y,x)
10	r(x,y),r(y,x),r(z,x)	r(x,x),s(x,x),t(y,x)	r(x,y),r(y,x),r(z,x)	r(x,y),r(y,x),r(z,x)

Table 4: The top ten cliques in each domain.

clause further improves the overall WPLL.

These parallels show a strong sibling resemblance between the two algorithms, and for this reason we do not claim that CSGL is a significant advance in the state of the art for structure learning, especially since LHL-FindPaths is already itself slow and unwieldy relative to full-fledged, lifted LHL (we were unable to directly compare CSGL with LHL because LHL currently has no public implementation). However, the similarities between CSGL and LHL-FindPaths do provide strong intuition for interpreting CSGL, and therefore DTM, as a structure learning algorithm.

One alternative to DTM for transfer learning in MLNs is TAMAR (Mihalkova, Huynh, and Mooney 2007), which discovers mappings between predicates in the source domain and predicates in the destination domain and then revises the MLN which is produced by the set of mapped clauses. Although TAMAR does explicitly use structure learning techniques to perform revision in the target domain, it outsources the process of learning in the source domain (its input is merely a set of source-domain clauses, which may be hand-constructed or learned from data using any algorithm) and so there is no obvious way to extend TAMAR to single-domain structure learning in the manner that we have shown is possible with DTM.

Conclusion and Future Work

We have shown that DTM can operate within a single domain to perform standalone structure learning, and in doing so perform as well or better than it does when incorporating knowledge from a separate source domain. Possible directions for future research include analysis of circumstances in which DTM might still benefit from cross-domain transfer (e.g. when the target domain has very little data), identifying other transfer learning mechanisms for which a similar self-transfer trick could be applied to improve single-task learning performance, and further exploring the connection between clique scoring and hypergraph pathfinding. Could clique scoring be integrated with LHL as a more sophisticated approach to identifying useful clauses?

Acknowledgements

Many thanks to Jesse Davis for graciously providing us with his implementation of DTM, the WebKB and Yeast Protein databases, and a great deal of helpful advice.

References

Biba, M.; Ferilli, S.; and Esposito, F. 2008. Structure learning of Markov logic networks through iterated local search. In *Proceedings of ECAI-2008*, 361–365. Amsterdam, The Netherlands: IOS Press.

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

Craven, M., and Slattery, S. 2001. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning* 43(1-2):97–119.

Davis, J., and Domingos, P. 2009. Deep transfer via second-order Markov logic. In *Proceedings of ICML-2009*.

Kok, S., and Domingos, P. 2005. Learning the structure of Markov logic networks. In *Proceedings of ICML-2005*, 441–448.

Kok, S., and Domingos, P. 2009. Learning Markov logic network structure via hypergraph lifting. In *Proceedings of ICML-2009*.

Kok, S.; Sumner, M.; Richardson, M.; Singla, P.; Poon, H.; Lowd, D.; Wang, J.; and Domingos, P. 2009. The Alchemy System for Statistical Relational AI. `http://alchemy.cs.washington.edu`. Technical Report, Department of Computer Science and Engineering, University of Washington, Seattle, WA.

Mewes, H. W.; Frishman, D.; Gildener, U.; Mannhaupt, G.; Mayer, K.; Mokrejs, M.; Morgenstern, B.; Minsterktter, M.; Rudd, S.; and Weil, B. 2002. Mips: a database for genomes and protein sequences. *Nucleic Acids Res* 30:31–34.

Mihalkova, L., and Mooney, R. J. 2007. Bottom-up learning of Markov logic network structure. In *Proceedings of ICML-2007*, 625–632. New York, NY, USA: ACM.

Mihalkova, L.; Huynh, T.; and Mooney, R. J. 2007. Mapping and revising Markov logic networks for transfer learning. In *Proceedings of AAAI-2007*, 608–614.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.

Torrey, L., and Shavlik, J. 2009. Transfer learning. In Soria, E.; Martin, J.; Magdalena, R.; Martinez, M.; and Serrano, A., eds., *Handbook of Research on Machine Learning Applications*. IGI Global.