Transfer Learning Across Relational and Uncertain Domains: A Language-Bias Approach

Raksha Kumaraswamy
School of Informatics & Computing
Indiana University
Bloomington, IN

Phillip Odom
School of Informatics & Computing
Indiana University
Bloomington, IN

Kristian Kersting
Dept. of Knowledge Discovery
Technical University of Dortmund
Germany

David Leake
School of Informatics & Computing
Indiana University
Bloomington, IN

Sriraam Natarajan
School of Informatics & Computing
Indiana University
Bloomington, IN

Abstract

Transfer learning is typically performed between problem instances within the same domain. We consider the problem of transferring across domains. To this effect, we adopt a probabilistic logic approach. First, our approach automatically identifies predicates in the target domain that are similar in their relational structure to predicates in the source domain. Second, it transfers the logic rules and learns the parameters of the transferred rules using target data. Finally, it refines the rules as necessary using theory refinement. Our experimental evidence supports that this transfer finds models as good or better than those found with state-of-the-art, and in a fraction of the time.

Introduction

Recent progress in machine learning has allowed for effective and accurate learning in the presence of large amounts of data. However, when the data set is small, learning performance may suffer. To alleviate this issue, *transfer learning* techniques (Pan and Yang 2010) were developed. In these methods, a source task is used for learning a model (or a set of models) that is then transferred to a related task where learning can be efficient given the bias of the source model. This technique has been successfully applied across several problems such as classification and sequential decision-making (Torrey *et al.* 2008; Raina *et al.* 2006; Mehta *et al.* 2008).

While successful, most of these techniques work with related problems and do not necessarily transfer across seemingly unrelated domains. To achieve this domain independent transfer, a richer representation such as first-order logic (FOL) is a minimal requirement (Davis and Domingos 2009; Haaren *et al.* 2015; Mihalkova *et al.* 2007; Mihalkova and Mooney 2009). As a motivating example, consider the NELL system (Carlson *et al.* 2010) that reads the web. Currently, NELL is well-versed in the sports domain, having learned several rich rules about sports organizations. To transfer the acquired knowledge to a different domain, say financial organizations, it is imperative to use a rich representation that allows modeling the objects, their relations and the uncertainty that inherently exists in both domains.

For learning from structured and uncertain domains, Statistical Relational Learning (SRL) (aka, Probabilistic Logic Models (PLM)) has been developed (Getoor and Taskar 2007). PLMs combine the richness of first-order logic with the ability of probability theory to handle uncertainty. Two different approaches have been applied for cross-domain transfer based on PLMs. The first set of methods (Davis and Domingos 2009; Haaren *et al.* 2015) employ second-order logic to model regularities between seemingly unrelated domains. The inherent assumption is that these methods possibly share a common sub-structure that can be exploited using higher-order logic. The second set of methods aim to find an explicit mapping of predicates using local search methods (Mihalkova *et al.* 2007; Mihalkova and Mooney 2009). Both these approaches employ the PLM of Markov logic networks (MLNs) to capture the source domain knowledge.

We follow the second approach of explicitly mapping the relational structure of the source to the target domains. For this purpose, inspired by the research in Inductive Logic Programming (ILP) (De Raedt and Kersting 2008), our approach performs "mode-matching" that compares the modes in both predicates[1]. Mode-matching compares the types of the arguments in the predicates of the source and target domain to identify potentially similar groups across the domains. Once the match is obtained, we perform a mode-based tree construction that allows us to construct the clauses in the target domain. This matching of the modes and the construction of the initial knowledge in the target domain can be seen as introduction of a *language-bias* for the target domain. Hence, this algorithm is called language-bias transfer learning (LTL).

Once the potential clauses are constructed, LTL employs two different types of refinements. The first is by exploiting the power of PLMs in allowing the *softening* of the rules. LTL learns new parameters for these rules from the training data in the target domain. We consider both weights (as with MLNs (Domingos and Lowd 2009)) and conditional probabilities (Kersting and De Raedt 2007) with combining rules (Natarajan *et al.* 2009). Some of the rules can be spu-

[1]Note the difference between modes in ILP and modes of probability distributions. Modes inside ILP define the argument types of a predicate and help in the inductive search of the rules

rious or inapplicable for the target, because they are simply constructed from predicate matching. Hence, softening them using training data allows for weighing down the rules that are not supported by data. As we show empirically, with less data, softening is an easier task than learning both the rules and the parameters. The second improvement is employing *theory refinement* (Ourston and Mooney 1990) by which LTL modifies some of the rules based on the training data. Specifically, it considers dropping a predicate or adding an attribute of one of the objects already present in the rule. For instance, if the rule mentions an organization and an employee, then our algorithm will search the space to potentially add an attribute of these two objects (say department) based on the improvement in the likelihood of the data.

Illustrative Example 1: Our language-bias transfer learning method relies on creating *mode-matched trees* (M^2Ts) in both the source (say UW-CSE (Richardson and Domingos 2006)) and target (say IMDb) domains. The simplified M^2Ts for the UW-CSE domain that models the relationship *advisedBy* is shown in Figure 1-*left*, and is generated from the rules in the source domain model. The rules correspond to predicting the advisor of each student, both of which are represented by the type *person*. The colors represent the parameter tying of the types in the clause, and the leaf nodes of the tree denote the rule - represented by a path from root to that leaf, if any, that has been learned or provided by an expert [2]. Based on the search process over these parameter types in the source, clauses with a similar structure are constructed in the target domain where the model predicts the relationship *workedUnder*. This is shown by Figure 1-*right*. Here, the rules predict if an actor has worked for a director, both of which are represented by the type *person*. As can be seen, for one source clause we get multiple target clauses because each source predicate is mapped to multiple target predicates, for example, *publication* is automatically mapped to *movie* and *genre*, as shown in the figure.

We make the following key contributions: First, we develop a language-bias based transfer learning algorithm (LTL) that allows for cross-domain transfer. Following the successes of several classical AI methods - ILP for predicate matching, SRL for modeling uncertainty in relational domains and theory refinement for improving background theories - we propose a transfer learning algorithm that leverages the benefits of all these methods. Second, we demonstrate the effectiveness and efficiency of our transfer learning algorithm in several real, complex tasks.

We proceed as follows: we first present the background necessary for the paper before outlining the key research ideas in the work, then we provide experimental results in several domains before concluding by outlining interesting research directions.

Related Work

The advantage of Probabilistic Logic Models (PLMs) (Getoor and Taskar 2007) is that they can succinctly represent probabilistic dependencies among the attributes of different related objects, leading to a compact representation of learned models. We consider two kinds of models in this work: undirected models that use weights and directed models that consider probability distributions.

One of the most popular PLMs is *Markov logic networks (MLNs)* (Domingos and Lowd 2009). An MLN consists of a set of formulas in first-order logic and their real-valued weights, $\{(w_i, f_i)\}$. Together with a set of constants, we can instantiate an MLN as a Markov network with a node for each ground predicate (atom) and a feature for each ground formula. All groundings of the same formula are assigned the same weight, leading to the following joint probability distribution over all atoms:$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right)$, where $n_i(x)$ is the number of times the ith formula is satisfied by a possible world x and Z is a normalization constant (as in Markov networks).

On the other end of the spectrum are directed models such as *Bayesian Logic Programs (BLPs)* (Kersting and De Raedt 2007) that employ conditional distributions for every clause (primarily horn clauses). The distributions, due to multiple instances of the same rules and multiple rules, are combined using combination functions (Natarajan *et al.* 2009) that combine multiple probability distributions into a single distribution. For the purposes of this work, it is sufficient to realize that the use of weights and probabilities are two different ways of softening the hard FOL clauses.

While there is significant research in learning these parameterized rules, especially for MLNs (Kok and Domingos 2010; Khot *et al.* 2011), these methods assume the presence of large amounts of training data. For learning with minimal data, the previously mentioned transfer learning methods that employ PLMs are closely related to our work. Specifically, the TAMAR algorithm (Mihalkova *et al.* 2007) and its extension SR2LR (Mihalkova and Mooney 2009) are quite similar in spirit. TAMAR maps a source MLN to a target MLN and uses a concept called *consistent-type* mapping which essentially maps one source type in a clause to one target concept. SR2LR on the other hand transfers short-range clauses from the source domain to develop longer-range clauses in the target domain. While these methods assume a MLN representation for the source, our method simply requires FOL clauses and the relational graph structure. Other algorithms such as DTM (Davis and Domingos 2009) and TODTLER (Haaren *et al.* 2015) use MLNs to create a second-order representation from the source that is then used to instantiate clauses in the target domain. While quite effective, these methods assume a hyperparameter that allows them to facilitate the transfer. We on the other hand, simply assume that the language bias of the target domain is obtained from the source domain and show that the data in the target domain can help in refining this bias to learn a more accurate model.

Transfer Learning using Language Bias

We first provide the intuition behind our approach using an example before providing the technical details. We considered the discriminative setting where we focus only on learn-

[2] Note that every path from the root to the leaf is a rule or clause in FOL. We use the words clauses and rules interchangeably.

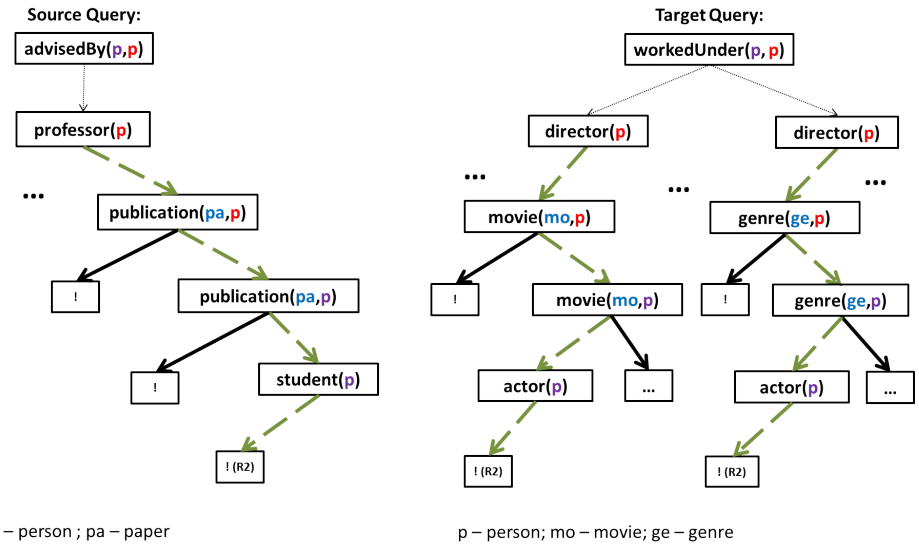

Figure 1: Examples of a simplified mode-based matching between the UW-CSE and IMDb datasets. It represents the flow of types which is shown with colors. Note that in the right figure, we present only a part of the search tree for brevity. Paths in the tree representing clauses are shown as shown as !(Ri).

ing horn clauses. For the rest of the paper, when we refer to a query, we refer to the head of the clause.

Given: Weighted (probabilistic) logic rules in the source domain, a small amount of training data, a set of queries and the type declarations in the target domain.

To Do: Learn a set of probabilistic clauses in the target domain for each query.

Illustrative Example: Our language-bias transfer learning method relies on creating *mode-matched trees* (M^2Ts) in both the source (say Yeast) and target (say WebKB) domains. The M^2T for the Yeast that models protein interaction, shown in Figure 2-left, is generated from the rules in the source domain model. The rules correspond to predicting the class of the protein. Each edge in the tree represents both the types that are shared with the query node as well as the number of variables that are shared across the connected nodes. The leaf nodes of the tree denote the rule (if any) that the path from root to the leaf represents. The use of "+" or "-" sign follows a typical ILP approach for the modes (type declaration of the variables). + denotes that the variable has already been defined in the query (correspondingly a - sign means that the variable is added to the body of the clause but is not present in its head).

The corresponding search tree for the WebKB target domain[3], shown in Figure 2-right represents all of the possible mode-matchings from the root node to any predicate in the target domain. Note that the example tree represents only a portion of the search tree for the WebKB domain. The key idea is that the query *deptOf* matches with *proteinClass* in the source. The predicate *interaction* has one type (denoted by $+p$) that matches with its query *proteinClass*. Correspondingly, in the target, the predicates *student*, *faculty* and *department* all match with its query *deptOf* (with a type $+wp$ present in both). Consequently, there can be several possible sub-trees of the query but we present only a part of the tree for brevity. The paths with the cuts (!) matching a rule in the source domain (for instance !(R1) and !(R2)), form the M^2T of the target domain.

Paths in the source domain will match paths in the target domain if the same number of arguments are related at each link in the path. For example, the red (dashed) lines and the green (long-dashed) lines show matched paths across the domains. There may be many such matched paths for every rule in the source domain. Intuitively, the bias that we are introducing in the target domain is the modes which are essentially a language bias as they restrict the search space inside the target domain. Another way to understand our approach is that, in a target domain search tree such as the one created by ILP learners (Ong *et al.* 2005), our method uses the mode-matches from the source domain to restrict the search inside the target search tree.

Approach: Modes and types are typically used in ILP systems to perform search efficiently. A mode in ILP typically refers to the definition of the types and the search bias inside a predicate. For instance, specifying that the *author* predicate (query) takes two arguments $\langle paper, person \rangle$ is defined as *author(+paper,+person)*. A "-" sign can be used as mode for a type when it is not present in the query but the introduction of its predicate can improve the search performance. For more details, we refer to the ILP book (De Raedt and Kersting 2008) but for the purposes of this paper, it is important to understand that modes are typically used as language bias for efficient search. More importantly, if declared correctly, they guarantee that the search procedure will terminate.

Definition 1 *M^2T_{node} - A node of an M^2T is a predicate*

[3]The goal here is to classify web pages

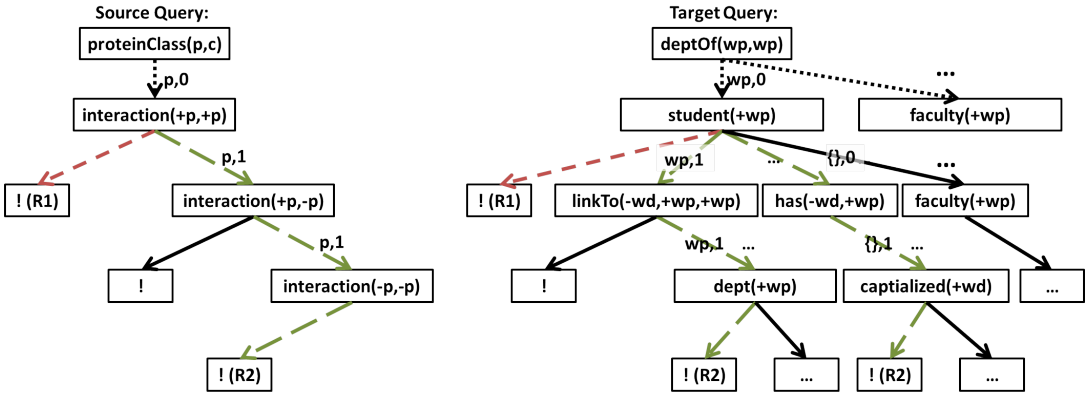

Figure 2: Examples of the mode-based matching between the yeast and webkb datasets. The source tree is built from two clauses (R1 and R2). The red dashed line and the green long-dashed line show corresponding paths across the source and target domains. Note that a single path in the source domain may "match" with many paths in the target domain. Note that in the right figure, we present only a part of the search tree for brevity. The final learned set of clauses are the paths that match with a particular rule (shown as !(Ri)).

with its corresponding modes defined for each argument.

Definition 2 M^2T_{edge} *- An edge of an* M^2T *is labeled with two parts. The first represents the modes in the lower-level* M^2T_{node} *that the edge is connected to that are shared with the query. The second represents the number of variables that are shared among the two* M^2T_{node}*s that are connected by this edge.*

Given the definitions of the nodes and edges, we now define our key data structure - M^2T.

Definition 3 M^2T *- A mode-matched tree (*M^2T*) is a tree rooted at the query* M^2T_{node} *and consists of* M^2T_{node}*s and* M^2T_{edge}*s.*

While the definition is simple, the origin of this tree is quite different in the source (M^2T_S) and target (M^2T_T) domains. For instance, for the source domain as shown in Figure 2-left, this is simply the knowledge/learned model. To be precise, this represents the set of clauses learned in the source but connected through their mode definitions. For the target domain, however, the tree presented on the right side of Figure 2 is a sub-set of the full search tree. M^2T_T, on the other hand (shown as the highlighted part of the right tree), is the result of applying the mode-matches to direct the search inside the full ILP search. Hence, for every source-target query pair, there will be two sets of trees - (source) M^2T_S which represents the set of clauses and (target) M^2T_T that represents the transferred set of clauses. M^2T_S is essentially used to bias the search in the search tree to construct M^2T_T. More precisely, the search process (which we explain next), uses the M^2T_S to build the M^2T_T from the potentially large ILP search tree.

Search Process: Algorithm 1 presents our Language-bias transfer learning (LTL) algorithm. Recall that every path in M^2T_S represents one clause in the source domain. The *GENMATCHES* function presents the search process. The function is called with every path in M^2T_S and the set of predicates ($\mathbf{P_T}$) in target domain. At a fairly high level, given the current source path, it constructs the set of M^2T_T paths that would form the target domain clauses. The key intuition is that these M^2T_T paths will have similar edge parameters compared to that of the input M^2T_S path.

More precisely, given the source path (*path*) and $\mathbf{P_T}$, the function traverses every node in *path* and finds similar matching nodes in the full target search tree to construct M^2T_T. To do so, the function compares the sequence of edge parameters in *path* with the sequence of edge parameters of every path in the full target tree. If there is a mismatch at any point in the sequence, the search is terminated for that target path. Hence, potentially for every *path*, several potential target clauses can be returned within M^2T_T for refinement in the next step. Note that LTL does not fully construct the search tree in the target domain, but incrementally constructs it and stops the search in that path when there is a mismatch in the modes between source and target domains. This allows it to learn efficiently compared to searching over all possible clauses.

Refinement of target clauses: We now have transferred a set of clauses to the target domain. For a single source clause, many target clauses may have been generated. For example, when transferring from Yeast to WebKB, 2 clauses in Yeast generate over 100 clauses in WebKB. This is due to the fact that *protein* potentially matches with *student*, *faculty*, *staff*, and *department* in WebKB. Also, Yeast has 5 predicates, while WebKB has nearly 15 predicates. Not all of these clauses capture true relationships in the target domain. Hence, these clauses need to be refined.

We consider two different types of refinements. First is the *softening* of these clauses through the use of weights or probabilities. Given a small amount of data in the target domain, we hypothesize that learning parameters will result in inaccurate clauses having low weights associated with them and accurate clauses having higher weights. We compare two different parameter learning approaches: Alchemy (Kok *et al.* 2007) weight learning for MLNs and combining rules (Natarajan *et al.* 2009) for BLPs. We used weighted mean for combining the instances due to different rules and

Algorithm 1 LTL: Language-bias Transfer Learning

function PERFORMTRANSFER($\mathbf{P_t}$,M^2T_{source},Q_S,Q_T)
  $\mathbf{rules} = \emptyset$
  for $path \in M^2T_{source}$ **do**
    $\mathbf{matches_T}$ = GENMATCHES($\mathbf{path}$,$\mathbf{P_t}$,Q_S,Q_T)
    $\mathbf{rules} = \mathbf{rules} \bigcup \mathbf{matches_T}$
  end for
  $\mathbf{rules}$ = REFINE($\mathbf{rules}$)
  return $\mathbf{rules}$
end function

function GENMATCHES($\mathbf{path}$,$\mathbf{P_t}$,Q_S,Q_T)
  $\mathbf{matches} = \emptyset$
  for all $node \in \mathbf{path}$ **do**
    $\mathbf{matches}' = \emptyset$
    for $p \in \mathbf{P_t}$ **do**
▷ Compare $node$ and source query argument types with p and the target query
      if EQUIV#TYPES($node$,Q_S,p,Q_T) **then**
        for all $m \in \mathbf{matches}$ **do**
▷ Compare p and $node$ variable matches with preceeding predicates
          if EQUIV#VARS($node$,p) **then**
            $\mathbf{matches}' = \mathbf{matches}' \bigcup m \wedge p$
          end if
        end for
      end if
    end for
    $\mathbf{matches} = \mathbf{matches}'$
  end for
end function

mean for combining the instances of the same rule. This approach was previously demonstrated to be effective (Natarajan *et al.* 2009) and hence we chose this combination over other popular combination functions such as Noisy-Or. We implemented gradient-descent with mean-squared error as the optimization criteria for learning the parameters. For weight learning in MLNs, we simply used Alchemy. In our implementation, we use the weights from the source clauses as initial parameters and refine them using data. It is possible to start with zero weights (or uniform distributions) and learn the new parameters. The two approaches did not yield significantly different results in our experiments.

The second type of refinement that we consider is the classic *theory refinement* (Ourston and Mooney 1990) where the key idea is to add or delete predicates in the clauses. This attempts to make the inaccurate clauses more applicable to the target domain. The significance of this refinement is that it allows the learning algorithm to build clauses that would otherwise not be found in the constrained search space because the properties of the M^2T would not match between the source and target. We use the improvement in likelihood as the criteria for adding a new predicate or deleting an existing predicate from the current clause. When adding a new predicate we consider only attributes of the current set of objects in the clause and do not add any new relations in the clause. Such refinement has been shown to be effective in PLMs (Natarajan *et al.* 2006). We present some sample transferred clauses in Table 2. For instance, in the Yeast domain, the *complex* predicate was dropped and only *function* and *locations* were used for predicting the class. In UW-CSE, an attribute of the Professor was modified.

UW-CSE ⇒ IMDb
S: prof(p) ⇒ advisedBy(s,p)
T: genre(per2, gen) ⇒ workedUnder(per1, per2)
S: pub(t, a) ∧ pub(t, b) ∧ prof(a) ∧ stud(b) ⇒ advisedBy(b, a)
T: mov(m, p1) ∧ mov(m, p2) ∧ act(p1) ∧ dir(p2) ⇒ workedUnder(p1, p2)
Yeast ⇒ WebKB
S: interact(a, a) ⇒ protein_class(a,c)
T: linkTo(w,wp1,wp2) ⇒ deptOf(wp1, wp2)
S: interact(c,c) ∧ interact(a,d) ∧ interact(c,a) ⇒ protein_class(a,b)
T: stud(wp1) ∧ dept(wp2) ∧ linkTo(wd, wp1, wp2) ⇒ deptOf(wp1,wp2)

Table 1: Sample clauses in a source (S) domain and corresponding transferred clauses from the target (T) domain.

UW-CSE
T: inPh(p1,ph) ∧ inPh(p2,ph) ∧ hasPos(p1,pos) ∧ prof(p1) ⇒ advisedBy(p2,p1)
R: stud(p2) ∧ inPh(p2,ph) ∧ hasPos(p1,pos) ∧ prof(p1) ⇒ advisedBy(p2,p1)
Yeast
T: comp(p,c) ∧ func(p,f) ∧ loc(p,l) ⇒ protein_class(p,c)
R: func(p,f) ∧ loc(p,l) ⇒ protein_class(p,c)

Table 2: A sample transferred clause (T) and refined clause (R) in two domains.

Illustration: To provide an idea of transferred rules, we present a few rules transferred from UW-CSE to IMDb data set and from Yeast to WebKB data set in Table 1. As can be seen, the first rule of UW-CSE states that if p is a professor, then $advisedby(p, s)$ is true i.e., p advises s. The corresponding rule in IMDb is that if $per2$ works in any genre, then $per2$ works under $per1$. A more interesting rule that maps nicely to a target rule is the following: if a student b writes a common paper with a professor a, then b is advised by a. The corresponding rule is that if an actor $p1$ works in the same movie directed by a director $p2$, then $p1$ works under $p2$. In the source rule, b and a are the types defined in the query predicate and the publication t is the new type introduced. Correspondingly in the target domain $per1$ and $per2$ are the two types present in query while the movie type mov is newly introduced. This similarity was obtained by traversing the respective M^2Ts of the source and target domains.

In summary, the high-level steps of our algorithm are presented in Figure 3. The source clauses are used to create M^2T_S. The target domain description (modes) allows us to create the possible search tree M^2T_T for each query predicate based on the language bias introduced from M^2T_S.

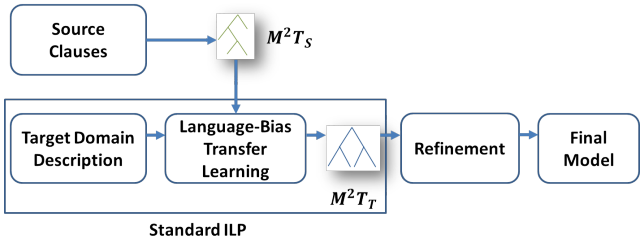

Figure 3: Our approach takes as input the source clauses and the target domain description and generates the M^2T trees. Then, our transfer learning approach creates clauses in the target domain, and refines them to output the final model.

Note that this is similar to the mode-directed path finding algorithm of Ong et al. [2005]. The resulting paths are then converted to clauses and then softened by learning probabilities from data. Finally, refinement (both probabilistic and theory) is performed to obtain a more accurate set of clauses for the target domain queries.

Experiments

In this section, we aim to investigate the following questions:

Q1: Does our method transfer efficiently across seemingly unrelated domains?

Q2: Does it generate accurate rules for the target domain?

Q3: Does *Theory Refinement* improve the performance?

Q4: How does our method compare against baselines?

In order to answer these questions, we consider 3 baseline approaches, which learn models from the minimal target data: (1) MLN structure learned using Alchemy (*MLN* in the results), and (2) *TAMAR* (Mihalkova *et al.* 2007), that performs cross-domain transfer using MLNs. The code of TODTLER (Haaren *et al.* 2015) and DTM (Davis and Domingos 2009) are not yet publicly available and comparison with these methods remains an interesting future direction.

For the clauses obtained using our *LTL* algorithm, we employ 2 methods for learning parameters: (1) weight learning of Alchemy (*LTL_WL* in the results), and (2) using weighted-mean as the combination function for these rules(Natarajan *et al.* 2009) (*LTL_CR* in the results). Similar to MLN structure learning, in MLN weight learning, we used the default settings of Alchemy to learn discriminative weights. In addition, LTL performed local theory refinement (added and dropped a predicate from the rules if needed) and learned parameters using combining rules (*LTL_CR_Ref* in the results).

We experimented with 2 pairs of data sets ((WebKB, Yeast protein) AND (Cora, IMDb)). In each pair (D_1, D_2) we transferred twice, by treating D_2 as the target and D_1 as the source, and vice-versa. Since the data sets contain different numbers of groups, we employed 4-fold cross-validation for the first transfer and 5-fold for the second.

To compare the performance of these various methods on the data sets, we use the following 5 measures: (1) conditional log likelihood (CLL), (2) mean squared-error (MSE), (3) area under the ROC curve (AUC-ROC), (4) area under the PR curve (AUC-PR) and (5) time in minutes. It is known that CLL in relational data sets can be misleading since the ratio of positive to negative examples is skewed. Predicting all the examples to be of the majority class can highly lead to confident yet misleading CLL values. Hence we use AUCs.

Table 3: Yeast $\Longrightarrow$ WebKB

METHOD	CLL	MSE	AUC-ROC	AUC-PR	Time
MLN	**-0.016**	**0.003**	0.501	0.004	147
TAMAR	-TO-	-TO-	-TO-	-TO-	-TO-
LTL_WL	**-0.016**	**0.003**	0.504	0.004	38
LTL_CR	-0.829	0.289	0.901	0.828	**26.074**
LTL_CR_Ref	-0.587	0.201	**0.953**	**0.921**	168.6

Table 4: WebKB $\Longrightarrow$ Yeast

METHOD	CLL	MSE	AUC-ROC	AUC-PR	Time
MLN	-0.059	**0.023**	**0.505**	0.027	0.117
TAMAR	**-0.051**	0.024	**0.505**	0.024	**0.083**
LTL_WL	-0.135	0.046	0.498	0.024	1.41
LTL_CR	-0.784	0.277	0.498	**0.378**	4.436
LTL_CR_Ref	-0.666	0.236	0.477	0.37	10.87

WebKB $\Longleftrightarrow$ Yeast protein: The WebKB dataset was first created by Craven et al.[2007] and contains information about department webpages and the links between them. It also contains the categories for each webpage and the words within each page. Some examples of predicates present are *student(webPage), linkTo(word,webPage,wordPage), samePerson(person,person)*. Here the goal is to predict the *departmentOf(webPage,webPage)* relation, which identifies a person as belonging to a particular department. The Yeast protein data set (Haaren *et al.* 2015) was obtained from the MIPS Comprehensive Yeast Genome Database(Mewes *et al.* 2002). The data set includes information about protein location, function, phenotype, class, and enzymes. Here the target is the *protein_class(protein,class)* predicate which classifies a protein to belong to a particular class.

Cora $\Longleftrightarrow$ IMDb: The Cora data set was first created by Andrew McCallum and later used by Bilenko et. al. (Bilenko and Mooney 2003). The goal in this is to predict the *samevenue(venue,venue)* relation which identifies two symbolic venue's of a conference as representing the same conference. The data set consists of details of of authors, their papers, and the venue the paper's are published at. The goal in the IMDb data set (Mihalkova and Mooney 2007), is to predict the *workedUnder(person,person)* relation which identifies an actor in the data set as having worked for a director. The data set consists of predicates with details like *actor(person), movie(movie, person), genre(movie,genre)* etc. and is split into 5 folds.

Results: Tables 3,4,5,6 present the results across the 4 transfer experiments. It can be observed from that tables that our methods (denoted as LTL_X) perform compara-

Table 5: Cora $\Longrightarrow$ IMDb

METHOD	CLL	MSE	AUC-ROC	AUC-PR	Time
MLN	-1.116	0.294	0.501	0.309	6
TAMAR	-0.846	0.254	0.501	0.3	7.334
LTL_WL	-1.937	0.289	0.83	0.769	7.01
LTL_CR	-0.317	0.102	**1.0**	**1.0**	**0.11**
LTL_CR_Ref	**-0.279**	**0.08**	**1.0**	**1.0**	**0.255**

Table 6: IMDb $\Longrightarrow$ Cora

METHOD	CLL	MSE	AUC-ROC	AUC-PR	Time
MLN	-1.876	0.324	0.59	0.505	**0.012**
TAMAR	-1.539	0.321	0.444	0.311	0.45
LTL_WL	-1.916	0.316	0.647	0.549	1.03
LTL_CR	-0.616	0.213	0.666	0.574	0.546
LTL_CR_Ref	**-0.612**	**0.211**	**0.678**	**0.585**	0.851

bly or better than the baselines in all the domains (**Q2**,**Q4**). TAMAR in particular, crashed after 24 hours in the WebKB domain with a memory exception as the number of predicates/clauses is significantly higher in this domain. MLN's default structure learning algorithm was not able to recover any useful structure in two of the four domains. TAMAR's learned MLN mostly consists of priors for all the target predicates. The lack of more informative clauses hurts the performance of TAMAR in most domains resulting in a poor AUC-ROC.

Comparing the different LTL_X approaches, it appears that the combination function based transfer was better than the weight learning approach in three domains. This could be due to Alchemy's default parameters not being sufficient to learn useful weights. The most surprising result was that performing local refinement of clauses (addition/deletion of a predicate) did not significantly improve the results compared to the probabilistic refinement(**Q3**). When we closely analyzed this in the different domains, we realized that the initial clauses obtained from the type matching themselves covered many of the useful clauses. We hypothesize that this is because we are discriminatively learning for some query predicates. When learning a joint model across several predicates, the refinement operation might prove to be crucial.

When comparing the timings, the combination function based transfer appears to be mostly faster than both Alchemy based learning methods (LTL_WL and MLN), and when it isn't better, it's performance is better. In the one case where MLN structure learning and TAMAR are faster (WebKB $\Longrightarrow$ Yeast), they do not learn any useful clauses and simply learn the priors (hence low AUC-PR). Hence **Q1** can be answered positively in that the proposed LTL methods are efficient when compared to the baselines.

Thus we can answer **Q1**, **Q2**, and **Q4** affirmatively while **Q3** needs investigation in more complex learning settings.

Conclusion

We presented a probabilistic logic approach for cross-domain transfer. Our proposed LTL algorithm builds on successes inside the ILP community to perform matching of modes (type declarations) in the source and target domains. These matches are then used as language-bias in the target domain to restrict the search over all possible clauses. Once the clauses are obtained in the target domain, the LTL algorithm refines them by softening them by learning parameters (weights or probabilities) and by performing local search. Our experimental results demonstrate that this method is both efficient and effective when compared to a rule learning algorithm and a recent cross-domain transfer algorithm (TAMAR). There are several possible interesting directions for future research. First is to extend the algorithm to generatively transfer the model of the entire domain. Second is to extend the theory refinement algorithms to consider broader global refinements than simple local ones that we considered in this work. Finally, incorporating human-advice in effectively guiding the transfer process remains a fascinating research direction.

Acknowledgment

SN gratefully acknowledges the support of the DARPA DEFT Program under the Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0039. KK acknowledges the German-Israeli Foundation (GIF) for Scientific Research and Development project 1180-218.6/2011. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, GIF or the US government.

References

M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, 2003.

A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka Jr, and T. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3, 2010.

J. Davis and P. Domingos. Deep transfer via second-order markov logic. In *Proceedings of the 26th annual international conference on machine learning*, pages 217–224. ACM, 2009.

L. De Raedt and K. Kersting. *Probabilistic inductive logic programming*. Springer, 2008.

P. Domingos and D. Lowd. Markov logic: An interface layer for artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–155, 2009.

L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

J. Haaren, A. Kolobov, and J. Davis. Todtler: Two-order-deep transfer learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

K. Kersting and L. De Raedt. Bayesian logic programming: Theory and tool. In *An Introduction to Statistical Relational Learning*, 2007.

T. Khot, S. Natarajan, K. Kersting, and J. Shavlik. Learning Markov logic networks via functional gradient boosting. In *ICDM*, pages 320–329, 2011.

S. Kok and P. Domingos. Statistical predicate invention. In *Proceedings of the 24th international conference on Machine learning*, pages 433–440. ACM, 2007.

S. Kok and P. Domingos. Learning markov logic networks using structural motifs. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 551–558, 2010.

S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, and P. Domingos. The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2007. http://alchemy.cs.washington.edu.

N. Mehta, S. Natarajan, P. Tadepalli, and A. Fern. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73:289–312, 2008.

H. Mewes, D. Frishman, U. Güldener, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Münsterkötter, S. Rudd, and B. Weil. Mips: a database for genomes and protein sequences. *Nucleic acids research*, 30(1):31–34, 2002.

L. Mihalkova and R. Mooney. Bottom-up learning of markov logic network structure. In *Proceedings of the 24th international conference on Machine learning*, pages 625–632. ACM, 2007.

L. Mihalkova and R. Mooney. Transfer learning from minimal target data by mapping across relational domains. In *IJCAI*, volume 9, pages 1163–1168, 2009.

L. Mihalkova, T. Huynh, and R. Mooney. Mapping and revising markov logic networks for transfer learning. In *AAAI*, volume 7, pages 608–614, 2007.

S. Natarajan, W. Wong, and P. Tadepalli. Structure refinement in first order conditional influence language. In *Proceedings of the ICML workshop on Open Problems in Statistical Relational Learning*, 2006.

S. Natarajan, P. Tadepalli, T. G Dietterich, and A. Fern. Learning first-order probabilistic models with combining rules. *AMAI*, 2009.

I. Ong, I. de Castro Dutra, D. Page, and V. Costa. Mode directed path finding. In *Machine Learning: ECML 2005*, pages 673–681. Springer, 2005.

D. Ourston and R. Mooney. Changing the rules: a comprehensive approach to theory refinement. In *AAAI*, pages 815–820, 1990.

S. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, Oct 2010.

R. Raina, A. Ng, and D. Koller. Constructing informative priors using transfer learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 713–720. ACM, 2006.

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

L. Torrey, J. Shavlik, T. Walker, and R. Maclin. Relational macros for transfer in reinforcement learning. In *Inductive Logic Programming*, pages 254–268. Springer, 2008.