Transfer Learning via Relational Type Matching

Raksha Kumaraswamy*, Phillip Odom*, Kristian Kersting†, David Leake* and Sriraam Natarajan*

*School of Informatics and Computing
Indiana University Bloomington

†Technical University of Dortmund
Dortmund, Germany

***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 method finds models as good or better than those found with state-of-the-art methods, with and without transfer, and in a fraction of the time.**

I. Introduction

Recent progress in machine learning and data mining 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 [1] 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 making learning efficient given the bias from the source model. While successful, most of these techniques work with related problems or within a single domain and do not necessarily transfer across unrelated domains. To achieve domain independent transfer, richer representations like relational models, structured representations like graphs or symbolic representations like first-order logic (FOL) is a minimal requirement [2]–[5]. Consider the NELL system [6] 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 [7]. 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 ([2], [3]) employ second-order logic to model regularities between seemingly unrelated domains. The inherent assumption is that these domains possibly share a common sub-structure that can be exploited using higher-order logic. The second set of methods ([4], [5]) aim to find an explicit mapping of predicates using local search methods. 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 domain. For this purpose, inspired by the research in Inductive Logic Programming (ILP) [8], our approach performs "type-matching" that compares the types, analogous to ILP's modes, between two predicates[1]. This helps identify potentially similar objects across the domains. Once the match is obtained, we perform a type-based tree construction to build the clauses in the target domain. This mapping of predicates based on types and construction of the initial knowledge in the target domain can be seen as introduction of a *language-bias* for the target domain. Therefore, this algorithm is called language-bias transfer learning (LTL). To handle incorrectness of the transferred clauses, LTL employs two different types of refinements.

In summary, 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 learning 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, LTL, that leverages the benefits of all these methods. Second, we demonstrate the effectiveness and efficiency of LTL in several real, complex and seemingly unrelated tasks.

II. Related Work

Probabilistic Logic Models (PLMs) employ FOL for representing complex structure, and probability theory for modeling uncertainty. The advantage of PLMs [7] is their capacity to succinctly represent probabilistic dependencies among the attributes of different related objects, leading to compact representations of learned models. We consider two kinds of models in this work: undirected models that use weights and directed models that use probability distributions.

One of the most popular PLMs is *Markov logic networks (MLNs)* [9]. 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).

[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.

On the other end of the spectrum are directed models such as *Bayesian Logic Programs (BLPs)* [10] that employ conditional distributions for every clause (primarily horn clauses). The distributions, due to multiple instances of the same rules and due to multiple rules, are combined using combination functions [11] 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 hard FOL clauses.

While there is significant research in learning these parameterized rules, especially for MLNs ([12]), these methods assume the presence of large amounts of training data. For learning with minimal data, previously mentioned transfer learning methods that employ PLMs are closely related to our work. Specifically, the TAMAR algorithm [4] and its extension SR2LR [5] are quite similar in spirit. TAMAR maps a source MLN to a target MLN using a concept called *consistent-type* mapping which essentially maps one source type to one target concept. SR2LR on the other hand transfers clauses with a small number of predicates (short-range clauses) from the source domain to develop longer-range clauses in the target domain. Other algorithms such as DTM [2] and TODTLER [3] 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 hyper-parameter that allows them to facilitate the transfer. In contrast, LTL requires source FOL clauses (which provide the language bias), and the target domain's relational structure. Our results show that target domain data can help refine this bias to learn a more accurate model.

III. Transfer Learning using Language Bias

We first provide a technical illustrative example before formally defining the approach. Here, we only consider the discriminative setting in which the task is learning rules that predict a particular query (i.e., horn clauses - rules of the form <**if** a *AND* b *AND* .. **then** q>, where "q" is the query). For the rest of the paper, when we refer to a query, we refer to the head of the clause.

Given: Weighted (probabilistic) logic horn clauses 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 horn clauses in the target domain for each query.

Illustrative Example: Figure 1 represents the transfer in a technical format for the transfer from Yeast →WebKB. The LTL method relies on creating *matching-type trees* (MT^2). The MT^2 for Yeast [3] that models protein interaction is shown in Figure 1-*left*. The rules correspond to predicting the class of the protein. Each edge in the tree represents ⟨types that are shared with the query node, number of variables that are shared across the connected nodes⟩. Again, 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 for types follows a typical ILP approach for type declarations, where + 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.

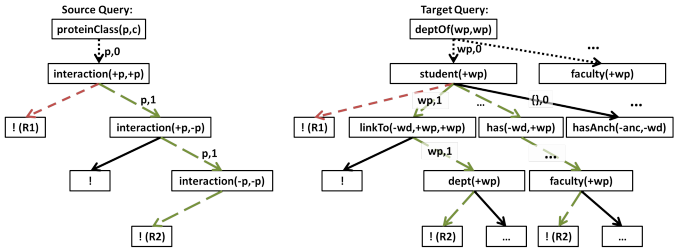


Fig. 1: Examples of the mode-based matching between the Yeast and WebKB datasets. The source tree is built from two clauses (R1 and R2). The red dashed line and the green long-dashed line show corresponding paths across the source and target domains. Note that a single path in the source domain may "match" 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)).

The corresponding search tree for the WebKB [13] target domain, where the goal is to classify web pages, is shown in Figure 1-*right* and represents all of the possible type-matchings from the root node to any predicate in the target domain. The key idea is that the target query *deptOf* matches with source query *proteinClass*. 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 (!) match a rule in the source domain (for instance !(R1) and !(R2)).

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 type-flow from the source which is essentially a language bias as it restricts 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 [14], LTL uses the type-matches from the source domain to restrict the search.

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 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 this predicate (that contains the new variable) can improve the search performance. More importantly, correctly declared modes guarantee that the search procedure will terminate. For more details, we refer to De Raedt and Kersting [8]. We adapt modes in this context to define constraint on the types. Given a clause each variable in the clause is assigned with "+" or "-" sign based on their occurrence. A new variable is assigned a "-" type when it is introduced and a "+" otherwise (i.e., when it is already present in the query). This assignment forms the constraint for the search tree in the target domain.

Algorithm 1 LTL: Language-bias Transfer Learning

function PERFORMTRANSFER($\mathbf{P_t}$,M^2T_{source},Q_S,Q_T)
  rules $= \emptyset$
  for $path \in M^2T_{source}$ **do**
    $\mathbf{matches_T}$ = GENMATCHES(**path**,$\mathbf{P_t}$,Q_S,Q_T)
    rules = **rules** $\bigcup \mathbf{matches_T}$
  end for
  rules = REFINE(**rules**)
  return rules
end function
function GENMATCHES(**path**,$\mathbf{P_t}$,Q_S,Q_T)
  matches $= \emptyset$
  for all $node \in$ **path do**
    matches$' = \emptyset$
    for $p \in \mathbf{P_t}$ **do**
▷ Compare $node$ and source query (Q_S) argument types with p and the target query (Q_T)
      if EQUIV#TYPES($node$,Q_S,p,Q_T) **then**
        for all $m \in$ **matches do**
▷ Compare p and $node$ variable matches with preceding predicates
          if EQUIV#VARS($node$,p) **then**
            matches$' =$ **matches**$' \bigcup m \wedge p$
          end if
        end for
      end if
    end for
    matches $=$ **matches**$'$
  end for
end function

Definition 1: MT^2_{node} - A node of an MT^2 is a predicate with its variables assigned as a "+"/"-" type for each argument.

Definition 2: MT^2_{edge} - An edge of an MT^2 is labeled with two parts. The first represents the types in the lower-level MT^2_{node} that the edge is connected to which are shared with the query. The second represents the number of variables that are shared among the two MT^2_{node}s connected by this edge.

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

Definition 3: MT^2 - A matching-type tree (MT^2) is a tree rooted at the query MT^2_{node} and consists of MT^2_{node}s and MT^2_{edge}s.

While the definition is simple, the origin of the two trees is quite different. For the source domain as shown in Figure 1-*left*, this is a model representing the set of clauses in the source connected through their type-flow definitions. For the target domain, however, the tree presented Figure 1-*right* is a sub-set of the full search tree. MT^2_T, the highlighted part of Figure 1-*right*, is the result of applying the type-flow constraints to direct the search inside the full search tree, thereby biasing it. Our hypothesis is that our approach can construct highly-predictive rules even with small amount of training data.

A. Search Process:

Algorithm 1 presents our language-bias transfer learning (LTL) algorithm's 2 parts. Recall that every path in MT^2_S represents one clause in the source domain. The *GenMatches* function presents the search process. This function is called for every path in MT^2_S by the *PerformTransfer* function with the set of predicates ($\mathbf{P_T}$) in the target domain. At a fairly high level, given the current source path, *GenMatches* constructs the set of MT^2_T paths that would form the target domain clauses. To do so, the function traverses every node in *path* and finds similar matching nodes in the full target search tree by comparing the sequence of edge parameters in *path* with the sequence of edge parameters along each path in the full target tree. If there is a mismatch at any point, the search along that target path is terminated. Potentially, for every *path*, several target clauses can be returned within MT^2_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 type constraints. This allows it to learn efficiently compared to searching over all possible clauses.

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

IMDb ⇒ Cora
S: mov(m, p1) ∧ mov(m, p2) ∧ act(p1) ∧ dir(p2) ⇒ workedUnder(p1, p2)
T: wordVen(v1,w1) ∧ wordVen(v2,w1) ∧ venue(p1,v2) ∧ wordVen(v2,w1) ⇒ sameVenue(v1, v2)
NELL: Sports ⇒ Finance
S: teamPlaysTeam(t1 ,t2) ∧ plays(s, t2) ⇒ teamPlaysSport(t1, s)
T: acquired(c1,c2) ∧ econSectorComp(s1, c2) ⇒ compEconSector(c1, s1)

B. Illustration:

To provide an idea of transferred rules, we present a few rules transferred from IMDb to Cora data set and from NELL-Sports to NELL-Finance data set in Table I. Please note that a rule, $a \wedge b \implies c$ can be read as <**if** a *AND* b **then** c>. An interesting rule that maps nicely to a target rule is the following: if a director d and an actor a worked in a movie m, then the actor a works for the director d. Similarly, if a word $w1$ appears in the venue $v1$ and venue $v2$, then $v1$ and $v2$ are the same venue. This was obtained by traversing the source and target domain MT^2s. The other two literals in the transferred clause are inconsequential because one is redundant while the other is satisfied for all venues. These irrelevant literals are removed during refinement, but it should be noted that their presence does not degrade performance in this case.

C. 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 be 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 the clauses capture true relationships in the target domain and hence need to be refined.

We consider two different types of refinements. First is the *softening* of these clauses through the use of weights

TABLE II: A sample transferred clause (T) and refined clause (R) in two domains.

Cora
T: wordVen(v1,w1) ∧ wordVen(v2,w1) ∧ venue(p1,v2) ∧ wordVen(v2,w1) ⇒ sameVenue(v1, v2)
R: wordVen(v1,w1) ∧ wordVen(v2,w1) ⇒ sameVenue(v1, v2)

NELL: Finance
T: bankInCountry(c2, country1) ∧ aquired(c2, c1) ⇒ compEconSector(c1, s1)
R: bankInCountry(c2, country1) ∧ aquired(c2, c1) ∧ econSectorComp(s1, c2)⇒ compEconSector(c1, s1)

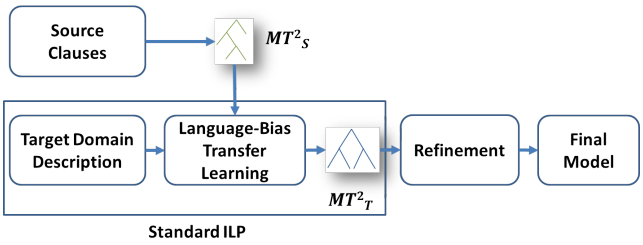


Fig. 2: Our approach takes as input the source clauses and the target domain description and generates the MT^2 trees. Then, our transfer learning approach creates clauses in the target domain, and refines them to output the final model.

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. We compare two different parameter learning approaches: Alchemy [15] weight learning for MLNs and combining rules [11] for BLPs. We used weighted mean for combining the instances due to different rules and mean for combining the instances of the same rule. This approach was previously demonstrated to be effective [11] and therefore 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 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 classic *theory refinement* [16] 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. We use the improvement in likelihood as the criteria for refinement. 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 [17]. We present some sample refined clauses in Table II. For instance, in Cora, a few predicates were dropped.

Summary: The high-level steps of our algorithm are presented in Figure 2. The source clauses are used to create MT^2_S. The target domain description (predicates) allows us to create the possible search tree MT^2_T for each query predicate based on the language-bias introduced from MT^2_S. Note that this is similar to the mode-directed path finding algorithm of Ong et al. [14]. The resulting paths are then converted to clauses. Finally, refinement (both probabilistic and theory) is performed to obtain a more accurate set of clauses.

IV. Experiments

We now aim to investigate the following questions:

- Q1 Does LTL transfer well across unrelated domains?
- Q2 Does LTL generate good rules for target domain?
- Q3 Does *Theory Refinement* improve performance?
- Q4 Does LTL perform better than the baselines?
- Q5 Does LTL perform better than the baselines with increasing amount of target data?

We consider 3 baseline approaches to learn from minimal target data: (1) MLN structure learned using Alchemy, (2) *TAMAR* [4] that performs cross-domain transfer using MLNs, and (3) *TODTLER*, which performs cross-domain transfer by lifting a source structure to second-order logic for transfer.

For the clauses obtained using our *LTL* algorithm, we employ 2 methods for learning parameters: (1) weight learning of Alchemy (*LTL_WL*), and (2) using weighted-mean as the combination function for these rules [11] (*LTL_CR*). 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*).

We experimented with 2 pairs of data sets ((WebKB, Yeast protein) and (Cora, IMDb)) and a dataset extracted from the NELL database [6] (sports domain → finance domain). 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, and 5-fold cross-validation respectively. For the NELL dataset, we split the data and performed 3-fold cross-validation.

It must be mentioned that to mimic the fact that we need to sometimes learn with a small amount of data in the target domain (and this is certainly true with the NELL data), we employed the following methodology: we split the data into n folds and learning is performed on 1 fold and tested on $n-1$ folds and this is repeated n times. This allows us to verify the hypothesis of learning from small datasets and shows the value of the bias introduced by transfer learning methods.

WebKB ⟺ Yeast protein: The WebKB dataset was first created by Craven et al. [13] 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)*. 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 [3] obtained from the MIPS Comprehensive Yeast Genome Database, includes

TABLE III: 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-
TODTLER	-2.578	0.189	0.509	0.001	75
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 IV: 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
TODTLER	-0.068	0.009	0.507	0.012	18.12
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

information about location, function etc. and the target is *proteinClass(protein,class)* that associates a protein to a class.

Cora $\Longleftrightarrow$ IMDb: The Cora data set was first created by Andrew McCallum and later used by Bilenko et. al. [18]. The goal in this is to predict the *samevenue(venue,venue)* relation which identifies two symbolic venues of a conference as representing the same conference. The data set consists of details of authors, their papers, and the venue the paper's are published at. The goal in the IMDb data set [19], 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..

NELL: Sport $\Longrightarrow$ Finance: NELL, an online never-ending machine learning system, has the ability to extract information from online text data, and convert this into a probabilistic knowledge base [6]. The data present in the knowledge base reflects the content of the web and varies with domain. Due to this, for some domains where the amount of data is less, there is a need for better reasoning algorithms to infer more probabilistic facts. LTL can facilitate learning rules in the target domain that has less data when knowledge from a source domain can be obtained (or even learned). We consider NELL data and transfer the knowledge rules from the *Sports* domain, where the task is to predict whether a team plays a sport, to learning in the *Finance* domain, where the task is to predict whether a company belongs to an economic sector.

Consolidated results: 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) transfer 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.

Results: Tables III, IV, V, VI and VII present the results across the 5 transfer experiments. It can be observed from these tables that our methods (denoted as LTL_X) perform comparably or better than the baselines in all the domains (**Q2**,**Q4**). TAMAR 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 five domains. The more recent TODTLER algorithm performs reasonably well on standard data sets and is comparable to our approaches in these domains. However, in the real world NELL data, it is significantly worse than our proposed algorithms. In 4 out of 5 domains, it needs order-of-magnitude more time than our approach.

TABLE V: 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
TODTLER	-0.417	0.116	0.944	0.924	358.287
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	0.087	1.0	1.0	15.295

TABLE VI: IMDb $\Longrightarrow$ Cora

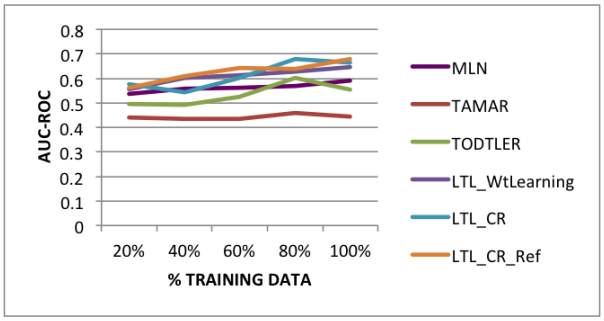
METHOD	CLL	MSE	AUC-ROC	AUC-PR	Time
MLN	-1.876	0.324	0.59	0.505	0.139
TAMAR	-1.539	0.321	0.444	0.311	0.45
TODTLER	-6.242	0.468	0.555	0.439	197.513
LTL_WL	-1.916	0.316	0.647	0.549	1.03
LTL_CR	-0.616	0.213	0.666	0.574	0.546
LTL_CR_Ref	-0.612	0.211	0.678	0.585	0.851

TABLE VII: NELL: Sports $\Longrightarrow$ Finance

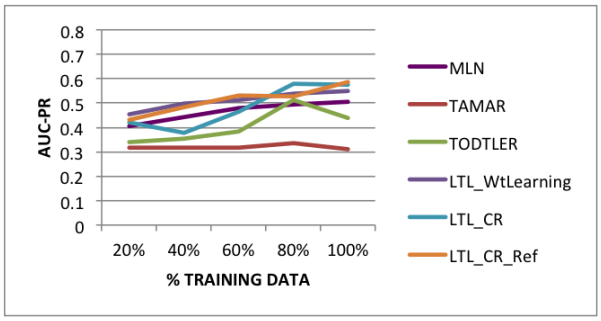
METHOD	CLL	MSE	AUC-ROC	AUC-PR	Time
MLN	-2.426	0.332	0.516	0.356	0.005
TAMAR	-1.139	0.306	0.518	0.378	22.96
TODTLER	-3.84	0.332	0.508	0.35	3168
LTL_WL	-2.571	0.331	0.513	0.387	15.62
LTL_CR	-0.642	0.224	0.597	0.422	1.056
LTL_CR_Ref	-0.618	0.214	0.7	0.518	1.9

Comparing the different LTL_X approaches, it appears that the combination function based transfer was better than the weight learning approach in two domains. This could be due to Alchemy's default weight learning parameters not being sufficient to learn useful weights. It can be observed that in several domains, performing local refinement of clauses significantly improved the results compared to the probabilistic refinement (**Q3**). This clearly shows that using mapping as *inductive bias* for learning in target domain improves performance. When comparing the timings, the combination function based transfer appears to always be faster than both Alchemy based learning methods (LTL_WL and MLN). In the one case where MLN structure learning and TAMAR are faster (WebKB $\Longrightarrow$ Yeast), they only learn priors (hence low AUC-PR). Hence, based on these experiments and observations **Q1** can be answered positively in that LTL transfers well across unrelated domains.

Learning curves: To compare the performance of our approach with increasing amounts of data and evaluate against the various baselines, we performed an additional transfer experiment (IMDb $\Longrightarrow$ Cora) by plotting a learning curve.



(a) AUC-ROC



(b) AUC-PR

Fig. 3: Learning curve comparing all algorithms: IMDb $\Longrightarrow$ Cora

In this domain, the standard deviation of all the algorithms is comparable, and therefore, it is used to ensure a fair comparison. For these experiments we compared two measures: (1) AUC-ROC, and (2) AUC-PR, where, each point of interest in the curve is obtained by averaging results from 20 runs.

Results: From Figure 3 it can see that at all the varying amounts of available training data, the LTL_X approaches outperform learning from scratch (MLN) and the other two transfer methods TAMAR and TODTLER (particularly with smaller amount of data). **Q5** can be answered affirmatively.

V. Conclusion

We presented a probabilistic logic approach for cross-domain transfer. Our LTL algorithm performs matching of type declarations in the source and target domains. These matches are 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 learning parameters (weights and probabilities), i.e., softening them, 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 interesting directions possible 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.

Acknowledgments: RK and SN gratefully acknowledge the support of the DARPA DEFT Program under the Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0039. PO and SN acknowledge the support of PARC XEROX through their faculty award. 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, ARO, AFRL, or the US government. We thank Jan Van Haaren and Jesse Davis for sharing their TODTLER code and data.

References

[1] S. Pan and Q. Yang, "A survey on transfer learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345–1359, Oct 2010.

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

[3] 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.

[4] L. Mihalkova, T. Huynh, and R. Mooney, "Mapping and revising markov logic networks for transfer learning," in *AAAI*, vol. 7, 2007, pp. 608–614.

[5] L. Mihalkova and R. Mooney, "Transfer learning from minimal target data by mapping across relational domains." in *IJCAI*, vol. 9, 2009, pp. 1163–1168.

[6] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka Jr, and T. Mitchell, "Toward an architecture for never-ending language learning." in *AAAI*, vol. 5, 2010, p. 3.

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

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

[9] P. Domingos and D. Lowd, "Markov logic: An interface layer for artificial intelligence," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, no. 1, pp. 1–155, 2009.

[10] K. Kersting and L. De Raedt, "Bayesian logic programming: Theory and tool," in *An Introduction to Statistical Relational Learning*, 2007.

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

[12] T. Khot, S. Natarajan, K. Kersting, and J. Shavlik, "Learning Markov logic networks via functional gradient boosting," in *ICDM*, 2011, pp. 320–329.

[13] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery, "Learning to extract symbolic knowledge from the world wide web," ser. AAAI, 1998, pp. 509–516.

[14] I. Ong, I. de Castro Dutra, D. Page, and V. Costa, "Mode directed path finding," in *Machine Learning: ECML 2005*. Springer, 2005, pp. 673–681.

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

[16] D. Ourston and R. Mooney, "Changing the rules: a comprehensive approach to theory refinement." in *AAAI*, 1990, pp. 815–820.

[17] 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.

[18] 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*, ser. KDD '03, 2003.

[19] L. Mihalkova and R. Mooney, "Bottom-up learning of markov logic network structure," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 625–632.