Interactive Transfer Learning in Relational Domains

Raksha Kumaraswamy, Nandini Ramanan, Phillip Odom & Sriraam Natarajan

ONLINE FIRST

KI - Künstliche Intelligenz
German Journal of Artificial Intelligence - Organ des Fachbereichs "Künstliche Intelligenz" der Gesellschaft für Informatik e.V.

ISSN 0933-1875

Künstl Intell
DOI 10.1007/s13218-020-00659-6

Springer

KI - Künstliche Intelligenz
https://doi.org/10.1007/s13218-020-00659-6

TECHNICAL CONTRIBUTION

Interactive Transfer Learning in Relational Domains

Raksha Kumaraswamy[1] · Nandini Ramanan[2] · Phillip Odom[3] · Sriraam Natarajan[2]

Received: 27 September 2019 / Accepted: 15 April 2020

Abstract
We consider the problem of interactive transfer learning where a human expert provides guidance to the transfer learning algorithm that aims to transfer knowledge from a source task to a target task. One of the salient features of our approach is that we consider cross-domain transfer, i.e., transfer of knowledge across unrelated domains. We present an intuitive interface that allows for an expert to refine the knowledge in target task based on his/her expertise. Our results show that such guided transfer can effectively reduce the search space thus improving the efficiency and effectiveness of the transfer process.

1 Introduction

Transfer learning [19] is a formalism inside machine learning that learns from a source task and transfers the knowledge (and possibly adapts/refines this knowledge) to a target task. This is extremely useful in the case where there is a paucity of training data in the target domain, enabling the exploitation of available knowledge to jump-start the learning process. 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 from the source model. This technique has been successfully applied across several problems including classification and sequential decision-making [12, 20, 22]. While successful, most of these techniques work with related problems or within a single domain and do not necessarily transfer across (seemingly) unrelated domains.

To achieve domain independent transfer, richer representations such as relational models, structured representation such as graphs or first-order logic (FOL) is a minimal requirement [4, 8, 13, 15]. As a motivating example, consider the NELL system [2] 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.

Two different approaches have been applied for cross-domain transfer based on first-order probabilistic methods. The first set of approaches [4, 8] 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 approaches [13, 15] aims to find an explicit mapping of predicates using local search methods. Both these approaches employ the probabilistic logic methods of Markov logic networks (MLNs) to capture the source domain knowledge.

Following the second approach, recently, we proposed a transfer method for relational data that uses search bias (language bias) from a source domain to accelerate learning in the target domain [11]. Inspired by the research in Inductive Logic Programming (ILP) [5], this approach performs "type-matching" that compares the types, analogous to ILP's modes, between two predicates.[1] Type-matching compares the types of the arguments in the predicates of the source and target domain to identify potentially similar objects across the domains. Once the match is obtained, we perform a type-based tree construction that allows our method to construct the clauses in the target domain. This matching of the predicates based on types and the construction of the initial

✉ Nandini Ramanan
Nandini.Ramanan@utdallas.edu

1 Computing Science, University of Alberta, Edmonton, Canada

2 Computer Science, University of Texas at Dallas, Richardson, USA

3 Georgia Tech Research Institute, Georgia Institute of Technology, Atlanta, USA

[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.

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).

While successful, LTL has a few significant issues: (1) it can potentially introduce hundreds of (probabilistic) rules in the target domain from a single source domain rule. In our prior work, the domain expert was used to refine the target domain rules . It is unreasonable to expect a domain expert to vet these large number of rules. (2) Many of these rules are potentially trivial/non-optimal w.r.t. the target domain data. Consequently, the burden on the expert in vetting out the rules *after learning* appears to be cumbersome. (3) The parameter tying aspect (where objects of similar types share the parameters) can be sub-optimal since the domains can be drastically different. There is a need to allow a human to interactively guide this process to learn optimal parameters.

To alleviate this issue, we introduce a novel *interactive transfer learning* approach that builds on the previous transfer method. The key difference w.r.t LTL is that we do not wait until the completion of the transfer process to solicit expert input. Instead, the expert interacts with the algorithm as the algorithm performs the search through the space of the rules. The expert has the ability to guide the search by pruning out seemingly inaccurate parts of the search space or could potentially introduce a new path for search or even instantiate particular variables of predicates as needed.

To facilitate seamless interaction, we also developed an interface that presents the expert with the current search process and accepts inputs from the expert in terms of adding a predicate, modifying it or suggesting a new branch to explore for the search. Given the interaction with the user, we develop a utility based heuristic method that chooses the optimal path to explore during transfer in the absence of expert input.

In this paper, we make the following contributions: (1) we introduce a novel framework that allows the expert to guide the transfer learning algorithm, (2) we define a robust utility based optimization problem that allows for efficient exploration in a new domain, (3) we design an intuitive interface that allows for seamless expert interaction, and (4) we empirically show that this method allows for much more efficient search than the original transfer method.

The rest of the paper is organized as follows—after reviewing the background on probabilistic logic models, we present the necessary details of LTL algorithm to make this work self-contained. We then discuss our interactive approach. Next, we present our empirical evaluation across four transfer tasks before concluding by outlining areas of future research.

2 Background and Prior Work

2.1 Background: Probabilistic Logic Models and Transfer Learning

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)* [6]. 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)* [9] 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 [16] 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 [10], 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 [13] and its extension SR2LR [15] 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 [4] and TODTLER [8] 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.

2.2 Prior Work: Language-Bias Transfer Learning

Our previous work [11] introduced a transfer learning approach (called LTL) using the language bias from the source domain to build candidate clauses (first-order rules) in the target domain of interest. The candidate clauses could then be scored based on the data and refined to generate more accurate clauses, hence accounting for differences between the source and target domains. Unlike previous work on transfer learning which used second-order logic to find patterns in the source to ground in the target domain [4, 8], the key idea is that LTL makes use of the basic search operations of Inductive Logic Programming (ILP) [5] to perform similar searches in the target domain as would be performed to build the source knowledge.

In ILP, given the definition of the input, we perform a greedy search through the space of possible theories, declaratively, by a set of mode definitions to guide the search process. Following the popular ILP system like ALEPH [21], we start with a bottom clause, variablize the ground statements and successively add/replace literals that optimizes the likelihood of a theory based on the data. Consider the following example where a professor is likely to coauthor with his/her students:

auth(per1,ppr1), auth(per2,ppr1), stud(per2) $\Rightarrow$ prof(per1)

In ILP, this source clause is built in the following steps, given that prof(per1) is the head:

- Add predicate with 1 argument matching with the head
- Add 2 predicates with 1 argument matching previous predicate

When transferring this to the problem of predicting the *boss* of a person, the following clause could be constructed:

proj(per1,proj1), proj(per2,proj1), employee(per2) $\Rightarrow$ boss(per1)

Notice how the structure (the flow of tied parameters) in the target domain clause is the same as in the source domain clause. While this clause seems reasonable, it is likely that many possible clauses exist in the target domain that satisfy the structure constraint imposed by a single source clause. LTL [11] represents the space of possible matches between source and target domain as a matching-type tree (MT^2). Following our original work [11], a MT^2 can be defined as,

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, a MT^2 can now be defined as,

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.

Figure 1-left, is the source tree, given as MT^2_S, represents the set of rules to predict advisedBy relation in the university domain.[2] For the target domain, the tree presented in Fig. 1-right is target search tree, given as MT^2_T with rules for workedUnder prediction. An example of this transfer from a university domain to a movie domain, as shown in Fig. 1, represents all of the possible type-matchings from the root node to any predicate in the target domain. The types refer to the possible parameter types that can be matched from the source clause to types in the target domain. The flow of instances of these types in a clause is often referred to as *parameter tying*, and this serves as the building block for the language-bias obtained from the source for transfer.

The key idea is that the query *workedUnder* in the target domain matches with query *advisedBy* in the source domain. The predicate *professor* has one argument with type (denoted by $+p$) that matches with its query *advisedBy*. Correspondingly, in the target domain, the predicate *director* matches with its query *workedUnder* (with a type $+p$ present in both). To predict the relationship *workedUnder* in the target domain, LTL searchs over these parameter types in the source domain to construct clauses with a similar structure in the target domain. For brevity, we present only a part of the MT^2_T tree on the right side of the figure. The paths with the cuts (!) in MT^2_T match a rule in MT^2_S of the source domain (for instance !(R2)). As shown in Fig. 2, at first LTL uses the source clauses to create MT^2_S. This MT^2_S acts as a language bias to create the possible search tree MT^2_T for each query predicate given the target domain description (predicates). This is similar in spirit to the mode-directed path finding algorithm of Ong et al. [18]. MT^2_T is then further pruned based on MT^2_S. Finally resulting paths in MT^2_T are converted to clauses. As part of refinement (both probabilistic and

[2] We use the subscripts S to denote the source domain and T, the target domain respectively.

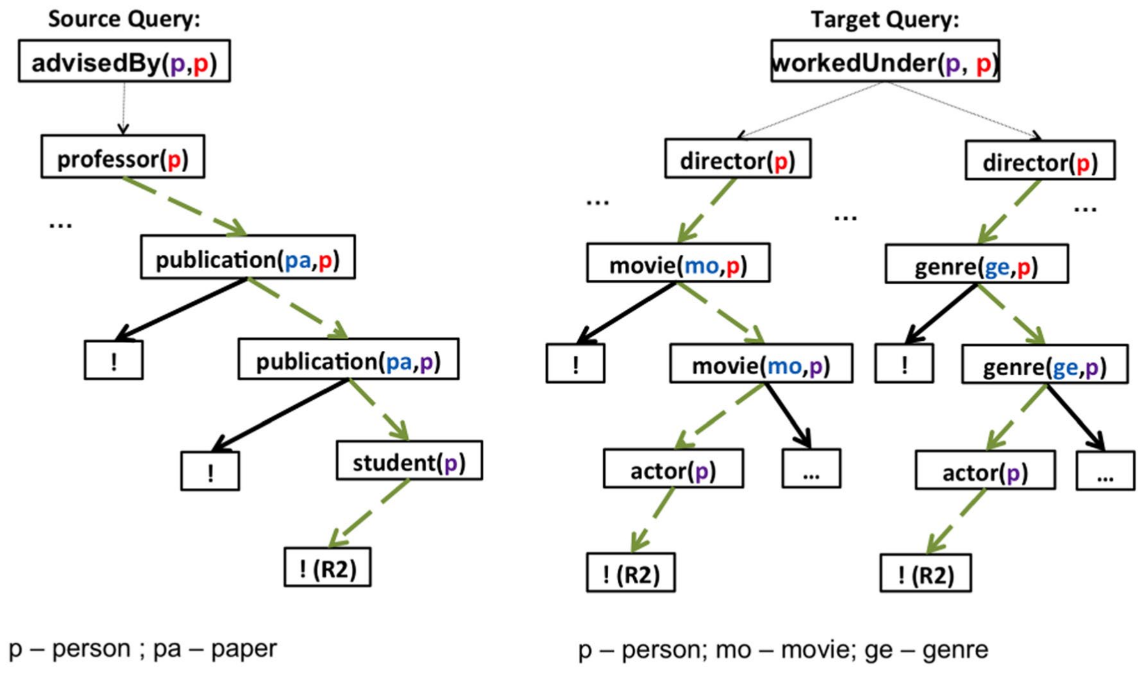


Fig. 1 Examples of a type-based matching between the UW-CSE and IMDb datasets. It represents the flow of types which is shown with colors. We present only a part of the search tree MT^2 for brevity. Paths in the tree representing clauses are shown as shown as !(Ri). Best viewed in color

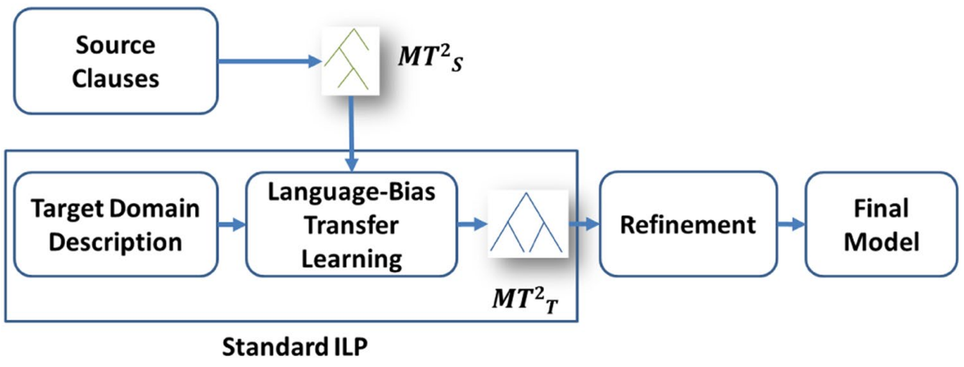


Fig. 2 `LTL` takes as input the source clauses and the target domain description and generates the MT^2 trees. Then, our transfer learning approach creates clauses in the target domain, and refines them to output the final model

theory), LTL can add or delete predicates in the clauses to obtain a more accurate set of clauses for the target domain queries.

Algorithms 1 and 2 present two primary components of Language-bias transfer learning. The *GenMatches* function in Algorithm 2 is called for every path in MT^2_S by the *PerformTransfer* function in Algorithm 1 to construct the set of MT^2_T paths that would form the target domain clauses, such that the MT^2_T paths will have similar edge parameters as that of the MT^2_S path. Note that LTL incrementally grows the search tree in the target domain and stops on type matching constraint violation thus improving the learning efficiency. Tables 1 and 2 present some sample transferred clauses and refined clauses respectively.

Algorithm 1 `LTL: Language-bias Transfer Learning - Transfer`

```
1: function PERFORMTRANSFER($\mathbf{P_t}$,$MT^2_S$,$Q_S$,$Q_T$)
2:     rules = $\emptyset$
3:     for path $\in MT^2_S$ do
4:         matches$_\mathbf{T}$ = GENMATCHES(path,$\mathbf{P_t}$,$Q_S$,$Q_T$)
5:         rules = rules $\bigcup$ matches$_\mathbf{T}$
6:     end for
7:     rules = REFINE(rules)
8:     return rules
9: end function
```

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

`IMDb` ⇒ `Cora`
S: director(d) ⇒ workedUnder(a,d)
T: venue(p1,v1) ⇒ samevenue(v1,v2)
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)
`WebKB` ⇒ `Yeast`
S: linkTo(w,wp1,wp2) ⇒ deptOf(wp1, wp2)
T: interact(a, a) ⇒ proteinClass(a,c)
S: stud(wp1) ∧ dept(wp2) ∧ linkTo(wd, wp1, wp2) ⇒ deptOf(wp1,wp2)
T: interact(c,c) ∧ interact(a,d) ∧ interact(c,a) ⇒ proteinClass(a,b)
`NELL: Sports` ⇒ `Finance`
S: teamPlaysTeam(t1 ,t2) ∧ plays(s, t2) ⇒ teamPlaysSport(t1, s)
T: acquired(c1,c2) ∧ econSectorComp(s1, c2) ⇒ compEconSector(c1, s1)
S: athletePlaysSport(a, s) ∧ onTeam(a, t) ⇒ teamPlaysSport(t1, s)
T: bankInCountry(c2, country1) ∧ aquired(c2, c1) ⇒ compEconSector(c1, s1)

Table 2 A sample transferred clause (T) and refined clause (R) in three 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)
`Yeast`
T: comp(p,c) ∧ func(p,f) ∧ loc(p,l) ⇒ proteinClass(p,c)
R: func(p,f) ∧ loc(p,l) ⇒ proteinClass(p,c)
`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)

Algorithm 2 `LTL: Language-bias Transfer Learning - Search`

```
1: function GENMATCHES(path,$\mathbf{P_t}$,$Q_S$,$Q_T$)
2:     matches = $\emptyset$
3:     for all $node \in$ path do
4:         matches$'$ = $\emptyset$
5:         for $p \in \mathbf{P_t}$ do
6:                 ▷ Compare $node$ and source query ($Q_S$) argument types with $p$ and the target query ($Q_T$)
7:             if EQUIV#TYPES($node$,$Q_S$,$p$,$Q_T$) then
8:                 for all $m \in$ matches do
9:                                 ▷ Compare $p$ and $node$ variable matches with preceding predicates
10:                    if EQUIV#VARS($node$,$p$) then
11:                        matches$'$ = matches$'$ $\bigcup m \wedge p$
12:                    end if
13:                end for
14:            end if
15:        end for
16:        matches = matches$'$
17:    end for
18:    return matches
19: end function
```

While these domains are clearly similar, our previous work [11] showed that the LTL algorithm is effective even in cases where the domains are not obviously similar. However, a key drawback of the approach is the difficulty in extracting the best clauses in the target domain, post transfer, especially when there is not sufficient target data to distinguish among the clauses. A single source clause can generate hundreds of matching target clauses in even moderately sized domains. We propose an interactive approach that is able to visualize the transfer process so that experts can assist the algorithm in selecting high-quality target clauses (or prune low-quality clauses) without having to define them a priori.

3 Guided Interactive Transfer Learning

To reiterate, our goal is to develop an *interactive learning* method that allows for effective transfer. Specifically, we are interested in leveraging the domain experts in the target domain to provide us with inductive/search bias that allows for *effective cross-domain transfer*. We now describe our interactive transfer learning algorithm. It consists of three fundamental components.

1. The heuristic adaptation of the transfer learning algorithm LTL, that we call as Heuristic LTL or *HLTL*
2. Set of interactions the expert performs in order to guide the algorithm; *Interactions*
3. The interface which enables this seamless interaction

We will explore each of these components serially in our path to constructing the framework. Before presenting each contribution, First, we will define the problem clearly.

Problem definition The goal is to generate clauses in target domain, K_T, given the clauses from source domain K_S. This is done by taking as input.

1. The possible add operations mapped from source to target (B)
2. The guidance provider/expert, E,
3. The small amount of target data available, D_T.

Formally, our goal is:

$$\begin{aligned} &\underset{b \in B, b_{params} \in b^{\#}}{\text{argmax}} \; w_E(b) \cdot P(lit_{target} \mid f(b, b_{params})) \\ B- &\quad \text{the possible branches at the current node} \\ b^{\#}- &\quad \text{the parameter tyings possible at the branch b} \\ f(b, b_{params})- &\quad \text{function to generate the clause body at node b,} \\ &\quad \text{with parameter tying } b_{params} \\ w_E(b)- &\quad \text{the weight assigned to a branch by the expert} \\ lit_{target}- &\quad \text{the target literal of the domain} \end{aligned} \tag{1}$$

$f(b, b_{params})$ is a function which generates the clause body at node b, by parsing the path to the node b, and instantiating node b to have b_{params}. We now explain this formulation along with the notion of confidence which determines the value of $P(lit_{target}|f(b, b_{params}))$ in this section.

3.1 Heuristic Based Language-Bias Transfer Learning

We adapt the LTL algorithm presented earlier as our base transfer algorithm for relational data. Recall that LTL performs transfer by utilizing the language-bias obtained from a single-piece of source knowledge to construct the MT^2 in the target domain. Each path in this MT^2 in the target represents a piece of transferred knowledge, and thus a single piece of knowledge from the source may result in an exponential number of clauses for target.

This behavior of LTL has three issues: (1) MT^2, as generated by the naive LTL algorithm during knowledge construction, represents huge amount of information concisely while compromising on real-time interpretability, (2) only a small fraction of this exponential set is representative of "good clauses" (those that perform well on the available target data), and (3) the small amount of available training data may be unrepresentative of the target domain, and LTL tries to overcome this problem by a stochastic refinement step which is not guided by experts.

Consequently, we introduce Heuristic based Language-Bias Transfer Learning (HLTL). In HLTL, we adapt LTL to include a utility heuristic (adapted from association rule mining literature and is essentially the likelihood) during its knowledge construction (building of MT^2). The likelihood of a clause (piece of knowledge denoted in FOL by $A \rightarrow B$, which is read as "A implies B") is computed using Bayes' rule as:

$$P(B \mid A) = \frac{P(A \wedge B)}{P(A)} \tag{2}$$

where, $P(X)$ is probability of X given the data. Our objective here is to maximize the likelihood (confidence/utility), since this signifies clauses which are validated by the available target domain data. HLTL utilizes this likelihood to help it improve upon LTL in two specific ways:

1. To perform the parameter tying for each generated piece of target knowledge
2. To help the algorithm pick the most promising branch for further exploration/knowledge construction based on the current candidate paths

The first way enables the expert E to see the most promising configuration of generated knowledge with respect to the available data. And the second one enables us to represent MT^2 construction in an exploitative (given the available data), and interpretable way.

Note that selecting a branch to explore involves considering the best parameter selection for that clause. As this can be expensive, we separate the choice of branch selection from the parameter selection step. This incremental construction of knowledge based on evaluating candidate branches, and exploring the most promising branch (in the absence of guidance), provides a constrained but optimal view of the knowledge transfer. Since this is an interactive framework, if the expert decides that the implicit behavior of the algorithm isn't optimal (due to noisy target data, restrictive constraints from the language bias etc.), then the expert can step in to provide guidance and change this implicitly wrong, albeit intelligent conditioned on current sources, behavior.

3.2 Types of Guidance

The optimality of the original LTL algorithm depends on available target data and the language bias from the source knowledge. Therefore, it is necessary to overcome any limiting effect of either of these two influences. Consequently, HLTL has a human-guided component. The natural question is, *what are the interactions necessary in order to overcome the limitations?* To answer this question, we first list the possible limitations.

1. Data available in the original/target can be noisy, and thus,

   - Parameter tying of a particular clause could be potentially wrong/sub-optimal (sub-optimal parameter tying). For instance, when transferring a rule about *coauthor(S1,P2) which models relationship between two authors to a domain* advisedBy(S2,P2) where $S2$ and $P2$ denote student and professor respectively (even though they are of type person at the highest level), one could incorrectly tie the parameters between S and P. This can lead to spurious rules as the behaviors of the two entities can be different in the target domain.
   - The branch taken to explore further during the incremental knowledge construction is sub-optimal (sub-optimal branch selection)

2. The language-bias from the source acts as a limiting factor and therefore HLTL is incapable of generating an unknown but effective piece of knowledge (sub-optimal knowledge generation). In essence, the *inductive bias* during transfer may limit the search space in the target domain. However, an efficient domain expert might posses the required knowledge. This knowledge could not have been considered (or rather explicitly avoided) by LTL since it could violate the language-bias constraint.

We now present three interactions to overcome these three limitations respectively. The new algorithm is presented in Algorithm 3.

- "Sub-optimal branch selection", can be addressed by providing the expert E with the ability to choose the branch to explore instead of or in addition to the the current best branch. This interaction is named *Explore Branch*. This interaction increases the weight assigned to a particular branch ($w_E(b)$) making it more likely to be explored. This is shown in lines 8–11 in Algorithm 3.
- The case of "sub-optimal parameter tying" can be addressed by providing the expert E with the ability to modify these tied parameters. Hence an interaction named *Modify Literal* is provided. This interaction alters $b^\#$ from Eq. 1. This is shown in lines 12–14 in Algorithm 3.
- For the case of "sub-optimal knowledge generation", the expert E needs to be able to add a path to MT^2_T that is being constructed. This can be done by adding an interaction called *Add Literal*. The result of this interaction changes the set of possible branches B. This is shown in line 25 in Algorithm 3.

Therefore our final set of interactions consists of {*Add Literal, Modify Literal, Explore Branch*} as the possible actions an expert E might need. Using these three basic interactions, as we show in our experiments, a domain expert will be able to guide HLTL to overcome its possible limitations and generate the best knowledge possible in the target domain, K_T, while utilizing the benefits of employing transfer learning to overcome the dearth of training data in the target domain.

Algorithm 3 HLTL - Search

```
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
                if $p$ != User_Preferred_Predicate then          } Explore
                    ▷ Explore a different literal                 } different branch.
                    continue;
                end if
                if CHECK_INCONSISTENT_PARAMS(node,$Q_S$,$p$,$Q_T$) then   } Inconsistent
                    MODIFYLITERAL(node,$Q_S$,$p$,$Q_T$)                   } parameter tying.
                end if
                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
    matches $\longleftarrow$ADDLITERAL(matches, $p \in \mathbf{P_t}$) hspace*8.5em }  Add New Literal.
    return matches
end function
```

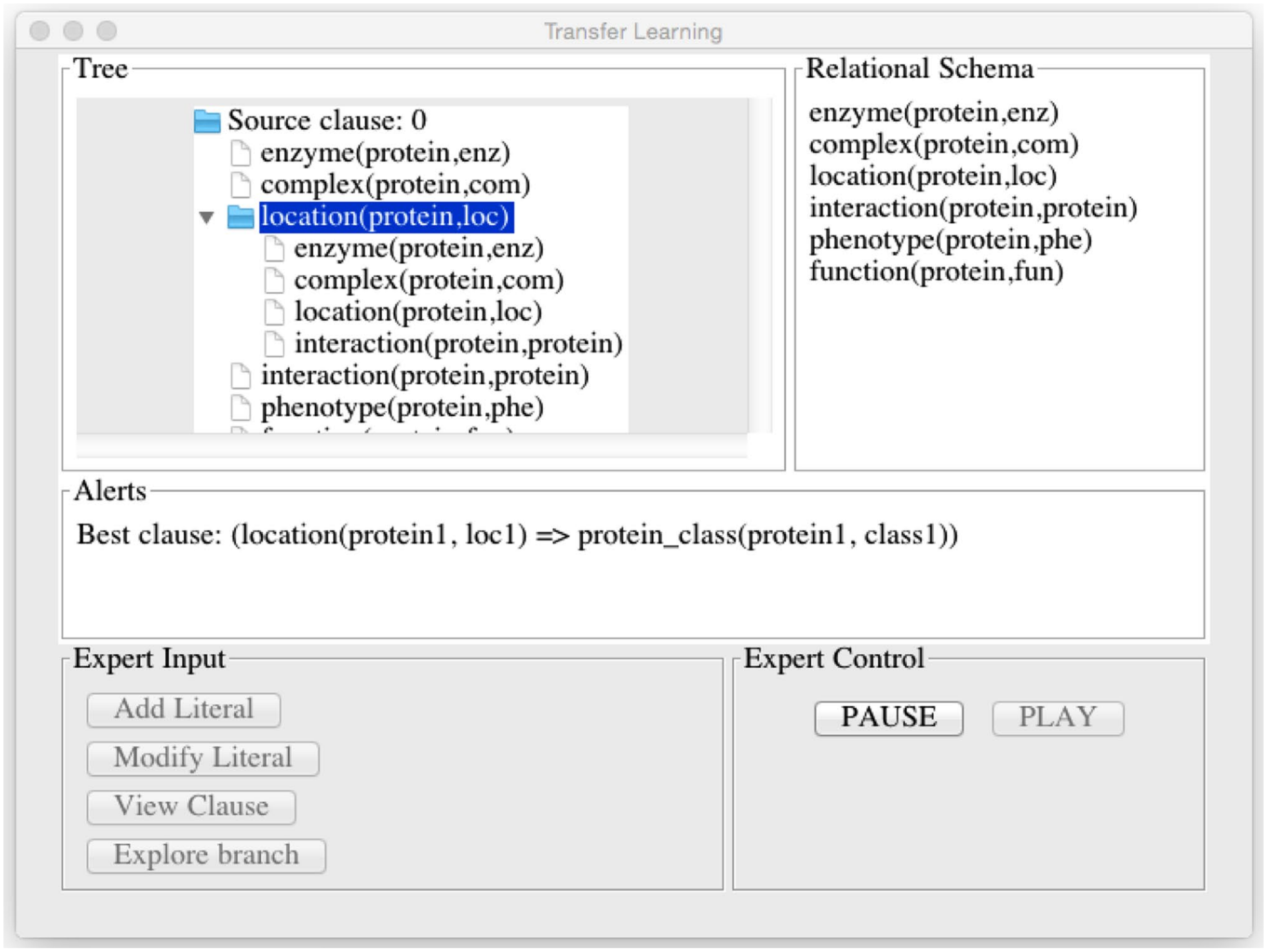
3.3 Interface for Guidance During Transfer

To facilitate the natural interaction with the expert who guides, the transfer, we built a user interface that we now discuss. Given the necessary knowledge about the underlying transfer algorithm employed, and the necessary interactions defined for guiding the transfer if necessary, let us now look at the interface that puts all this together to enable this interaction in a seamless manner. The layout of the interface is shown in Fig. 3 and has *three* functional components stacked up.

1. The top component is the "visualizer" which consists of 2 parts: (1) the left panel, *Tree*, to show the MT^2 construction in real-time, and (2) the right panel, *Relational Schema*, which shows the ontology of the target domain. The primary purpose of this component, is to facilitate the expert E visualize the current model, and the possible alternatives/modifications to it (through the ontology).
2. The middle component is the "informer" aka *Alerts*. This component enables HLTL to convey, in the form of messages (strings), any information it needs, to the expert E.
3. The bottom component is the "controller". This component again consists of two parts: (1) the right panel, *Expert Control* consists of buttons which enable the expert E to control the execution of HLTL (by Pausing/Playing it) in order to interact with it, and (2) the left component, *Expert Input*, enables the expert E to carry out interactions depending on their choice of interaction from the *Interactions* set. The left component is dynamic, in that, its temporal contents are determined by the choice of interaction; and in a *paused* state of the transfer, multiple interactions can occur before the user explicitly chooses to stop interacting (by selecting to *Play*), or if the "interaction time" as predefined in the settings of execution is exceeded (this ensures that an unresponsive expert E does not cause the process to be inefficient).

The interface, particularly the view showing the construction of the MT^2 tree has been designed/implemented to make intuitive and real-time understanding of the knowledge construction process to the expert E. There are two important features of this interface.

1. *View clause utility* Relational knowledge consists of the domain definitions, and a set of parameter tyings among these selected predicates. The parameters reflect meaningful instances of the combination and hence, it is important to ensure these tyings are optimal. In our interface, the expert has an option to *View clause* at any node in the MT^2. It is important to note that this is not a part of the *Interactions* set because this particular action does not convey any message from the expert to HLTL. This feature just enables the expert to view the clause at



Best clause: (location(protein1, loc1) => protein_class(protein1, class1))

Fig. 3 A snapshot of the interface for *HLTL*

any node and add this clause to the final set of clauses if deemed necessary by the expert.

2. *Parameter tying propagation* A feature of any tree that makes it intuitive is the concept of the *parent/child* relationship. To preserve this intuition, once the parameter tyings of a node has been determined at the n^{th} level of knowledge construction, the exploration of possible parameter combinations at its child is constrained with respect to the parent's tied parameters configuration. This ensures that all children of the same parent have similar configurations (preserving the meaning of a parent node). If the expert determines that the configuration of a parent is sub-optimal and changes it by *Modify Literal* action, this modification is propagated down the sub-branch below this node. The expert can alternatively add a new node to the tree, which is the same literal, but whose parameters are tied as preferred. This utilizes expert efficiently while preserving the sovereignty of being a *parent*, and enables the expert to decide the granularity of their input.

Given our description of the interactive LTL method, we now turn our attention to evaluation.

4 Experiments

Our experiments will aim to answer the following questions in order to demonstrate the benefits of HLTL.

Q1 How does HLTL's performance compare to LTL?
Q2 Does HLTL improve LTL's time efficiency?
Q3 How much effort does the expert *E* spend in guiding the transfer process?

4.1 Experimental Setup

We now compare our HLTL approach against LTL by utilizing the same framework used in the transfer experiments [11]. Unlike the general practice of *Machine Learning*, we

Table 3 Learning time for LTL and HLTL in seconds

Method	WebKB→Yeast	Cora→IMDb	IMDb→Cora	NELL:Sports→Finance
LTL	652.20	15.30	51.09	114.00
HLTL	2.47	15.26	5.55	3.36

Table 4 Effort spent by expert

Method	WebKB→Yeast	Cora→IMDb	IMDb→Cora	NELL:Sports→Finance
HLTL + expert	2	11	9	4
Only expert	15	36	15	12

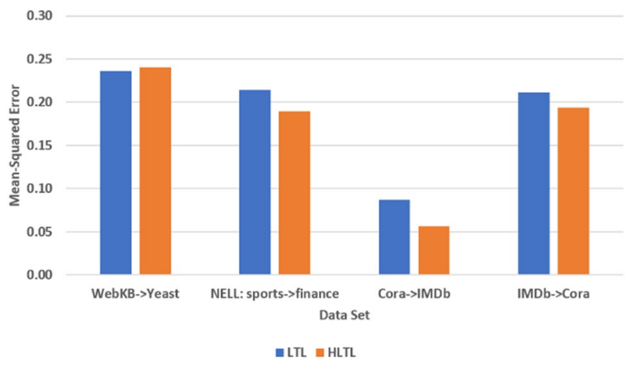


Fig. 4 MSE plot for the experiments. The comparisons clearly demonstrate that the proposed HLTL method is comparable to LTL method on minimizing the error

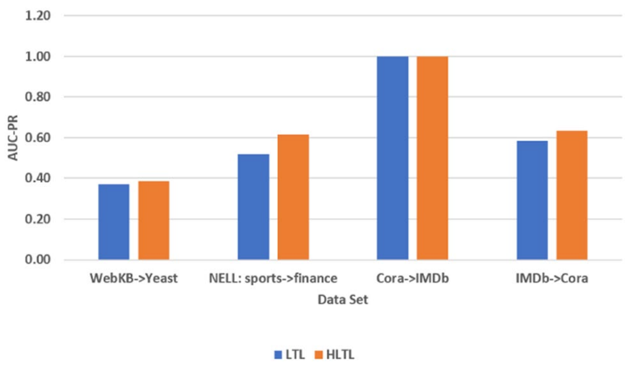


Fig. 6 AUC-PR plot for the experiments. The comparisons clearly show that the proposed HLTL method is comparable to LTL method in AUC-PR

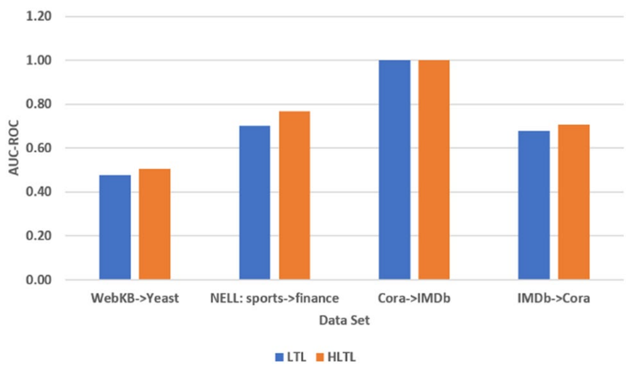


Fig. 5 AUC-ROC plot for the experiments.The results clearly demonstrate that the proposed HLTL method is comparable to LTL method on AUC-ROC

perform 5-fold, 4-fold, and 3-fold cross-validation for each of the transfers respectively with 1-fold for training and (n − 1) folds for testing, as with the original paper. This is carried out for all the combinations of the n folds, as in the classical interpretation. This allows us to confirm the hypothesis of learning from small data sets and demonstrated the value of the bias introduced by transfer learning methods.

4.2 Domains

We experimented with three pairs of data sets (WebKB ⟺ Yeast protein) and (Cora ⟺ IMDb) and (sports domain ⟺ finance domain) extracted from the NELL database [2].

We perform five transfer tasks as listed below.

Cora ⟺ IMDb The Cora data set was first created by Andrew McCallum and later used by Bilenko et al. [1]. Cora consists of research publications information, and the goal is to predict samevenue(venue, venue) i.e., if two instances of venues refer to the same conference given author, title, venue, haswordauthor, haswordtitle, haswordvenue etc. IMDB data set proposed by Mihalkova and Mooney [14], contains predicates like Actor, Director, Movie, Genre etc. The task in IMDb data set is to predict who works under whom WorkedUnder(person, person).

WebKB ⟹ Yeast protein The WebKB data set created by Craven et al. [3] and later converted by Mihalkova and Mooney [14] to contain the category of each web-page in a university framework and links between these pages. The task here is to predict the particular department of a webpage departmentOf(webPage, webPage) given predicate like Student, samePerson, linkTo etc. from these webpages. The Yeast protein data set [8] obtained from the MIPS Comprehensive Yeast Genome Database, includes information about location, function etc. and the target is *proteinClass(protein,class)* that associates a protein to a class. The Yeast protein data set [8] obtained from the MIPS Comprehensive Yeast Genome Database, includes information about proteins. The task is to classify these proteins given proteinClass(protein, class) f unction, location etc.

NELL:Sport $\Longrightarrow$ Finance We consider NELL, an online never-ending machine learning system [2]. NELL has the ability to extract information from online text data, and convert this into a probabilistic knowledge base. Here, we consider the task of transferring knowledge from *Sports* domain (which consists of a lot of data in NELL, and therefore robust knowledge), where the task is to predict whether a team plays a sport, to a *Finance* domain. The target goal is to predict the economic sector of a company (data collected by web-crawling is minimal).

To compare the performance of these various methods on the data sets, we use the following three measures: (1) mean squared-error (MSE), (2) area under the ROC curve (AUC-ROC), and (3) area under the PR curve (AUC-PR). A plot for each of these performance measures is shown in Figs. 4, 5, and 6 respectively. Across all the domains one can observe that HLTL achieves a lower MSE and comparable or higher AUC ROC and PR. Thus it can be stated that HLTL's over all performance is comparable or better than LTL, answering **Q1** affirmatively.

While the predictive performances are similar, our hypotheses was that this interactive learning will significantly reduce the learning time. This can be observed by comparing the learning times for LTL and HLTL as shown in Table 3. A key observation is that HLTL is superior in all the domains. In three of the fpur domains this difference is significantly high (in the order of 10×–250×). It can be concluded that HLTL uses the expert interaction efficiently to perform transfer in a fraction of LTL's learning time, thus answering **Q2** positively.

While the learning time can be significantly reduced, this can be wasteful if the expert's time is significantly high. To evaluate the amount of effort spent by expert *E*, we compare the number of interactions an expert performs to completely guide the learning algorithm, minus the transfer. These results in Table 4 show that constructing knowledge in the target in this case can be expensive in terms of expert effort; and therefore to answer **Q3**, the expert spends minimal effort when acting as a guide to the transfer process. Please note that another potential drawback of not employing transfer can be the lack of input to the expert, since any intuition available from the data can no longer be exploited.

In summary, it can be concluded that *HLTL significantly reduces the training time and the effort of the expert without sacrificing the performance of the learning algorithm.*

5 Conclusion

We considered the problem of "deep" transfer where the goal is to learn to transfer across seemingly unrelated domains. To this effect, we built upon our previous work which introduced bias in the search using the language bias from the source domain. Specifically, we developed an interactive tool that allows for a domain expert to seamlessly interact with a transfer learning algorithm. This tool builds on deep transfer method and demonstrates superior efficiency compared to the original method. Most importantly, it is clear from our experiments that the expert's efforts significantly decrease when learning interactively.

There are a few directions to pursue in future. First, we plan to rigorously evaluate this tool in different domains. In our current approach, the user intervenes and provides the necessary inputs. A more interesting direction would be to allow the learning agent [17] to query the user actively. Allowing for richer forms of human inputs including but not limited to preferences, qualitative constraints and even partial models may accelerate learning. Extending the tool to allow for richer modalities of inputs such as natural language text, gestures and direct manipulation of models etc can increase the interactive nature of the solution. Finally, going beyond transfer learning to allow for sequential decision making and temporal reasoning tasks/models remains an interesting direction.

Acknowledgments SN gratefully acknowledges the support of CwC Program Contract W911NF-15- 1-0461 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO). SN & NR gratefully acknowledge AFOSR award FA9550-18-1-0462. Any opinions, findings and conclusion or recommendations are those of the authors and do not necessarily reflect the view of the DARPA, ARO, AFOSR or the US government.

References

1. Bilenko M, Mooney R (2003) Adaptive duplicate detection using learnable string similarity measures. In: ACM SIGKDD
2. Carlson A, Betteridge J, Kisiel B, Settles B, Hruschka E Jr, Mitchell T (2010) Toward an architecture for never-ending language learning. In: AAAI
3. Craven M, DiPasquo D, Freitag D, McCallum A, Mitchell T, Nigam K, Slattery S (1998) Learning to extract symbolic knowledge from the world wide web. AAAI
4. Davis J, Domingos P (2009) Deep transfer via second-order markov logic. In: ICML

5. De Raedt L, Frasconi P, Kersting K, Muggleton S (2008) Probabilistic inductive logic programming. Springer, New York
6. Domingos P, Lowd D (2009) Markov logic: an interface layer for artificial intelligence. Synthesis lectures on artificial intelligence and machine learning
7. Getoor L, Taskar B (2007) Introduction to statistical relational learning. MIT Press, Cambridge
8. Haaren J, Kolobov A, Davis J (2015) Todtler: two-order-deep transfer learning. In: AAAI
9. Kersting K, De Raedt L (2001) Bayesian logic programs. arXiv:cs/0111058
10. Khot T, Natarajan S, Kersting K, Shavlik J (2011) Learning Markov logic networks via functional gradient boosting. In: ICDM
11. Kumaraswamy R, Odom P, Kersting K, Leake D, Natarajan S (2015) Transfer learning via relational type matching. In: ICDM
12. Mehta N, Natarajan S, Tadepalli P, Fern A (2008) Transfer in variable-reward hierarchical reinforcement learning. Machine Learning
13. Mihalkova L, Huynh T, Mooney R (2007) Mapping and revising markov logic networks for transfer learning. In: AAAI
14. Mihalkova L, Mooney R (2007) Bottom-up learning of markov logic network structure. In: ICML
15. Mihalkova L, Mooney R (2009) Transfer learning from minimal target data by mapping across relational domains. In: IJCAI
16. Natarajan S, Tadepalli P, Dietterich TG, Fern A (2009) Learning first-order probabilistic models with combining rules. AMAI
17. Odom P, Natarajan S (2016) Actively interacting with experts: A probabilistic logic approach. In: Joint European conference on machine learning and knowledge discovery in databases. Springer
18. Ong IM, de Castro Dutra I, Page D, Costa VS (2005) Mode directed path finding. In: Gama J, Camacho R, Brazdil PB, Jorge AM, Torgo L (eds) Machine learning: ECML 2005. Lecture notes in computer science, vol 3720. Springer, Berlin, Heidelberg
19. Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE transactions on knowledge and data engineering
20. Raina R, Ng A, Koller D (2006) Constructing informative priors using transfer learning. In: Proceedings of the 23rd international conference on Machine learning. ACM
21. Srinivasan A (2007) The aleph manual
22. Torrey L, Shavlik J, Walker T, Maclin R (2008) Relational macros for transfer in reinforcement learning. In: Blockeel H, Ramon J, Shavlik J, Tadepalli P (eds) Inductive logic programming. ILP 2007. Lecture notes in computer science, vol 4894. Springer, Berlin, Heidelberg