# Learning through Advice-Seeking via Transfer

Phillip Odom[1], Raksha Kumaraswamy[1], Kristian Kersting[2], and Sriraam Natarajan[1]

[1] Indiana University, Bloomington, Indiana
phodom,rakkumar,natarasr@indiana.edu
[2] Technical University of Dortmund, Germany
kristian.kersting@cs.tu-dortmund.de

**Abstract.** Experts possess vast knowledge that is typically ignored by standard machine learning methods. This rich, relational knowledge can be utilized to learn more robust models especially in the presence of noisy and incomplete training data. Such experts are often domain but not machine learning experts. Thus, deciding what knowledge to provide is a difficult problem. Our goal is to improve the human-machine interaction by providing the expert with a machine-generated bias that can be refined by the expert as necessary. To this effect, we propose using transfer learning, leveraging knowledge in alternative domains, to guide the expert to give useful advice. This knowledge is captured in the form of first-order logic horn clauses. We demonstrate empirically the value of the transferred knowledge, as well as the contribution of the expert in providing initial knowledge, plus revising and directing the use of the transferred knowledge.

## 1 Introduction

There has been an increased interest in building intelligent agents with a human-in-the-loop. This interest has been partially fueled by the rapid development of advice-taking systems [21, 7, 13] that do not rely merely on data but utilize domain advice provided by the expert. While specific adaptations differ, these systems are motivated by the fact that there has been decades of knowledge acquired by experts in various fields and restricting them to be "mere labelers" places undue importance on possibly noisy data while ignoring their expertise.

In this work, we consider the formalism of probabilistic logic (PL) [8] for learning from rich, structured, and possibly noisy data. Previously, a knowledge-based PL learning approach was proposed [18] that adapted a powerful boosting algorithm [16] to accept human advice about specific regions of the feature/state space to learn in structured domains. It uses a pre-defined set of human expert rules as advice in every iteration of the boosting algorithm to ensure the learned model is robust even in the presence of significantly noisy data.

While successful, this method assumes that the expert provides all relevant advice in advance. This increases the burden on the expert significantly. While in classical systems the burden on the expert was to generate examples/labels, in

knowledge-based systems the burden shifts to providing relevant advice. We aim to lessen this burden by providing the expert with an initial set of *bias* (advice rules) that the expert can modify/adapt based on their knowledge.

Inspired by a recently successful transfer learning technique that identified similarities across seemingly unrelated domains [12], we propose to employ transfer learning for providing the initial bias to the human expert. The key idea in our approach, that we call *learning through advice-seeking via transfer* (LAST) is to transfer knowledge from a source domain to generate a set of potential *advice rules* in the target domain. Then, these advice rules are provided to a domain expert who could potentially refine the current set of rules. In turn, these rules can then serve as advice for the subsequent learning algorithm.

Consider providing advice to a system that predicts the advisor of a student. The current knowledge-based PL system [18] requires the domain expert to provide rules such as "students co-author papers with their advisors", "students TA for their advisor's courses" etc. However, assume that we have knowledge in a different domain like movies, where we have rules such as "actors work in movies with directors", "actors and directors typically work in similar genres" etc., to predict if an actor works with a director. Now, using the transfer learning approach, we can potentially map actors to students, directors to advisors, movies to papers, genres to departments and create a set of potentially interesting rules that can be refined by the expert in the target domain. These refined rules can then be combined with (noisy) data to learn a robust model. This can significantly reduce the burden on the expert.

This paper makes the following key contributions – (1) It proposes the first transfer-based approach for advice-giving to learning algorithms. (2) It combines a successful advice-taking PL approach and a transfer learning approach in a seamless manner. (3) It reduces the burden on the domain expert by automatically identifying relevant rules and restricts the expert input to simple refinement operations. (4) Finally, it demonstrates excellent empirical performance in several benchmark data sets and on a large real-world Never Ending Language Learning (NELL) task [3].

## 2 Background

Learning through advice-seeking via transfer is related to both transfer learning and knowledge-based probabilistic logic learning.

### 2.1 Transfer Learning

Recently, there has been an increasing interest in the development of techniques that leverage information from a possibly related task to accelerate learning in the current task. Collectively called *transfer learning* [19], they learn a model for a source task, and transfer/adapt this learned model to a potentially related and similar target task. Transfer learning has been explored previously in the context of cognitive science [9, 11]. Transfer learning methods that transfer across

seemingly unrelated domains can be divided into two groups - the first group consists of methods that assume that the two domains share an underlying relational structure, even though they may appear dissimilar. Consequently, these methods employed higher order logic to model this structural similarity [10]. Alternatively, the second set of methods search for explicit mappings between the two domains and transfer rules from the source accordingly [14].

We consider a relational type-matching [12] transfer method called "language-bias transfer learning" (LTL) that uses type matching typically done in Inductive Logic Programming [5]. LTL utilizes types of arguments to map source predicates to target predicates, identifying similar objects in the two domains, which are then used to construct clauses in the target domain. This approach was shown to obtain state-of-the-art results in PL domains.

Inspired by this, we propose the use of this transfer method for generating good domain knowledge in the target domain that can then potentially be refined by an expert. Such a generation has two major advantages. First, it reduces the burden on the expert to generate several advice clauses in the target domain that can be used for learning. Second, it improves the results of the LTL method because it allows for the model transferred by the algorithm to be used to correct (possibly noisy) data in the target domain. We now explain the background of the learning method that can effectively exploit domain advice when learning with noisy data.

### 2.2 Knowledge-based Probabilistic Logic Learning

Previous work [17] extended standard functional gradient boosting [6] to learning relational models. The key intuition to functional gradient boosting is to transform the problem of discriminatively learning a large, complex model into a series of smaller, simpler problems. This is accomplished by learning a series of models—each one capturing some of the error w.r.t. the current model. In relational function gradient boosting, each step of the learning problem is to learn a single relational regression tree (RRT) [2]—binary decision trees with first-order logic atoms in the nodes and regression values in the leaves.

While shown to be effective across a wide variety of different problems, Relational Functional-Gradient Boosting (RFGB) requires high-quality data in order to learn a good model. Recently, there has been work on using knowledge-based learning to learn in the presence of noisy data in relational domains [18]. This introduced a knowledge-based framework that used label preferences to target and correct noisy data. It assumes that experts will have the appropriate knowledge of the domain to identify areas where noise is likely in the training data. The learning bias is specified in the form of first-order logic clauses ($\wedge f_i(x) -> advice(x)$). The body ($\wedge f_i$) specifies a set of logical conditions that define the set of examples to which the advice will apply while the head ($advice(x)$) of the clause defines the label preferences. This can be written as a tuple $< \wedge f_i, \mathbf{Pref}_{label}, \mathbf{Avoid}_{label} >$.

Label preferences are a natural way for the expert to communicate. For instance, when considering heart attacks, an expert might say "people who have a family history of heart attacks are more likely to have a heart attack than a

stroke". Here the body of the clause specifies that a patient's family member has had a heart attack and the head of the clause has heart attack as a preferred label to stroke.

While this framework performs well in the presence of systematic noise, it places an unnecessary burden on the expert. This is because, a key assumption is that the human expert will be able to identify the most informative advice (i.e., that the expert will know where the systematic noise is present in the data). This assumption can fail in several scenarios—for example, with domain experts who may not inspect historical data on a regular basis.

The solution that we present next is to employ transfer learning for generating the body of the advice, i.e., defining for which examples the algorithm is interested in having label preferences. The expert could then simply refine this advice based on his/her expert knowledge.

## 3 Advice-Seeking for Transfer

Our goal in this work is to facilitate a natural human interaction with the learning algorithm by allowing for the human to be involved in several stages—as an expert providing (minimal) knowledge in the source domain and as an expert who can potentially look at several clauses in the target domain and vet out or refine the clauses in the target domain.

Consequently, our *Learning through Advice-Seeking for Transfer* (LAST) algorithm generates relevant advice clauses for the target domain from expert-provided clauses in a source domain. This advice serves as a recommendation for useful advice to the expert. While the expert is still responsible for providing the label preferences for the advice, his/her task is simplified through transfer learning. Thus, significant effort is reduced for the expert. In turn, this makes the system more cognitively aware, i.e., it thinks like a human in developing prior knowledge. It is key that the transfer algorithm is able to generate appropriate (in this case relevant, or near-relevant) knowledge in the target domain. As a motivating example, consider providing advice in the case of a movie domain.

**Illustrative Example:** In order to learn on a movie domain, let us assume the presence of knowledge in an university domain. This domain comprises of faculty, students, the publications that they author, and the courses they are involved in. Networks like these are common across universities. Assume that the knowledge provided in this domain aims to predict the advisor of a student. One such knowledge could be that the students are more likely to co-author with their advisor (as against with a random professor in their department).

$$paper(p1, per1), paper(p1, per2), student(per2) \Rightarrow advisor(per1, per2)$$

The goal is to use this knowledge to transfer to a movie domain (say imdb) with movies, actors and directors where the target is to predict which actors have worked under which directors. When transferring the clause from the academic domain to the movie domain, the language bias approach [12] will produce many

different predictive rules. Let us consider one such clause

$$mov(m1, per1), mov(m1, per2), act(per1) \Rightarrow workedunder(per1, per2)$$

Notice how this clause captures a relationship, as actors do work under another person working on the same movie. However, this clause also covers actors working under actors in the same movie. If this rule is provided to the expert who understands the domain, he/she might suggest different refinements to this rule: 1). He/she could suggest that actors work under directors by adding a predicate to the clause. 2) Alternatively, he/she might suggest that actors do not work under each other. These lead to the following prior knowledge that can be used by the algorithm of Odom et al [18].

$$1) < [mov(m1, per1), mov(m1, per2), act(per1), dir(per2)],$$
$$workedunder(per1, per2), \neg workedunder(per1, per2) >$$
$$2) < [mov(m1, per1), mov(m1, per2), act(per1), act(per2)],$$
$$\neg workedunder(per1, per2), workedunder(per1, per2) >$$

### 3.1 The Problem Formulation

We now formally define our problem:

| | |
|---|---|
| *Given* : | Source knowledge, noisy training data, and access to an expert |
| *Todo* : | 1) Transfer knowledge from the source to target, |
| | 2) Solicit advice about this knowledge from expert |

The goal of the advice seeking problem is to select the transferred knowledge about which to query the expert (denoted $\mathbf{q} \subseteq \mathbf{K}$). Useful knowledge in our context is a measure ($\delta$) that is a function of the performance (say accuracy) on the training data ($\mathbf{D}$). While the goal is to maximize the performance of the queries on the training data, there is a cost ($C$) for making a query to the expert (as the expert has a limited budget to provide advice). Hence, the goal is $arg\max_{\mathbf{q} \subseteq \mathbf{K}} \delta_{\mathbf{q}}(\mathbf{D}) - C(\mathbf{q})$. We make the simplifying assumption that the algorithm can select $n$ queries to ask the expert. We also assume that there is a constant cost for every query. It is an interesting direction to personalize these costs based on difficulty of each query.

As mentioned earlier, we employ a relational transfer learning approach to generate knowledge ($\mathbf{K}$) in the target domain [12]. The set of queries can then be selected from the potential list of clauses by considering the performance of each clause on the training set. While the training set does suffer from noise, the hypothesis is that reasonable knowledge will still be generated and the expert can refine the knowledge into the appropriate advice.

## 4 The LAST Algorithm

We will now describe the components of our LAST algorithm (shown as algorithm 1). The algorithm takes as input the noisy training data, the descriptions

of the source and target domains, a set of knowledge about the source domain and a domain expert to query. The overall goal of LAST is to generate a set of knowledge about the target domain (which we refer to as advice) and use it to learn a more robust model in the target domain. The algorithm proceeds as follows:

### 4.1 Step 1: LTL

First, we transfer the set of knowledge in the source domain to the target using the LTL [12] algorithm. This transfer learning technique leverages the structure of the knowledge in the source domain to generate knowledge with similar structure in the target domain (making use of the domain descriptions). While effective, LTL generates all possible similar rules resulting in an impractical number of potential expert queries. The next step mitigates this issue.

### 4.2 Step 2: SelectBestN

We select from among the large potential set of queries by comparing their accuracy on the training set. Selecting the top $N$ clauses allows the algorithm to control the number of queries that can be directed to the expert. While more advice benefits the learning algorithm, this parameter can be tuned based on the availability of the expert.

### 4.3 Step 3: Improve

While LTL can generate appropriate knowledge in the target domain, it does not account for noisy training data. Therefore, the expert can modify the knowledge as needed to correct for any differences between the source and target domains. As this can be a time consuming process, the expert is not required to correct all (or any) of the knowledge. As we show empirically, sometimes minimal refinement is all that is needed to improve the learning algorithm. In several cases, the use of a powerful learning algorithm that can exploit the provided advice as bias can be expected to perform reasonably well in the target domain with noisy data.

### 4.4 Step 4: Query

Now that appropriate knowledge has been provided in the target, the expert is queried about the label preferences. As mentioned in the background, advice consists of three components: the features describing to which examples the advice will apply, the preferred (more likely) labels for those states, and the avoided (less likely) labels for those states. The previous step defined the features while this step solicits the label preferences. Now the advice is fully defined.

### 4.5 Step 5: Learn

The final step learns a robust model from the noisy training data and the expert advice. We use an advice-based learning algorithm called KBPLL [18] which is able to effectively utilize the expert advice to learn in the presence of noisy training data. It learns by trading-off between the training data and any advice. A key advantage of this approach is that it can not only learn with any amount of advice, but it is also capable of handling conflicting advice.

**Algorithm 1** The LAST algorithm: Learning through Advice-Seeking via Transfer

**Data**, domain description in the source domain ($\mathbf{D}_S$), domain description in the target domain ($\mathbf{D}_T$), expert ($E$), knowledge in the source domain ($\mathbf{K}_S$)
**function** LAST($\mathbf{Data}, \mathbf{D}_S, \mathbf{D}_T, \mathbf{K}_S, E$)
  $\mathbf{K}_T$ = LTL($\mathbf{Data}, \mathbf{D}_S, \mathbf{K}_S, \mathbf{D}_T$)
  $\mathbf{q} \in \mathbf{K}_T$ = SelectBestN($N, \mathbf{K}_T, \mathbf{Data}$)
  $\mathbf{q}_r$ = Improve($E, \mathbf{q}$)
  $\mathbf{A}$ =Query($E, \mathbf{q}_r$)
  **return** Learn($\mathbf{A}, \mathbf{Data}$)
**end function**

Algorithm 1 presents these different steps. Please recall that the expert is involved in both designing the source clauses and potentially refining the target clauses.

## 5 Experimental Evaluation

We aim to address the following questions:

**Q1**: How effectively does LAST utilize the transferred advice?
**Q2**: How important is the contribution of the expert?
**Q3**: Does LAST properly assist the expert to generate advice?
**Q4**: What is the quality of the advice generated?

### 5.1 Domains Used

We use four standard PL domains – imdb, cora, uw and webkb – and the large-scale NELL domain. In each domain we incorporate random noise (20%) to demonstrate that our algorithm is capable of building robust models. The domains are introduced pairwise - the source/target domain for transfer.

The **imdb** domain [15] consists of information about actors, directors and movies. The overall goal of this domain is to predict which actors work under which directors. This domain is paired with the **cora** domain [1] which consists of information about the details of author, their publications and venues. The

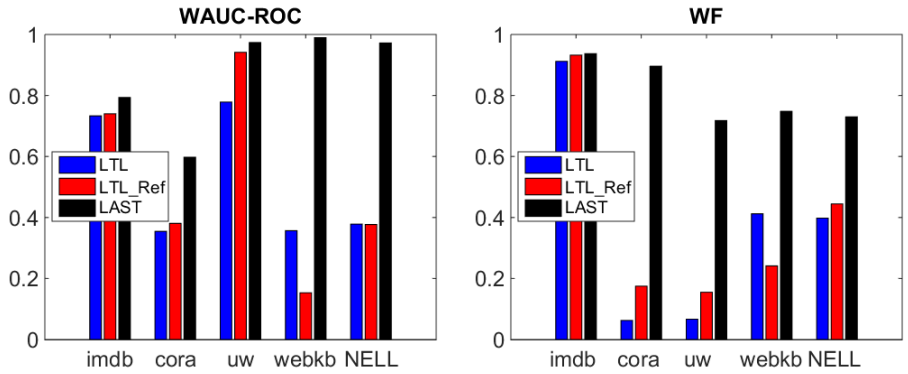


**Fig. 1.** Experimental results in each of our domains. We compare both weighted-AUC ROC (**LEFT**) and weighted-F score (**RIGHT**) in each domain. Higher bars represent improved performance.

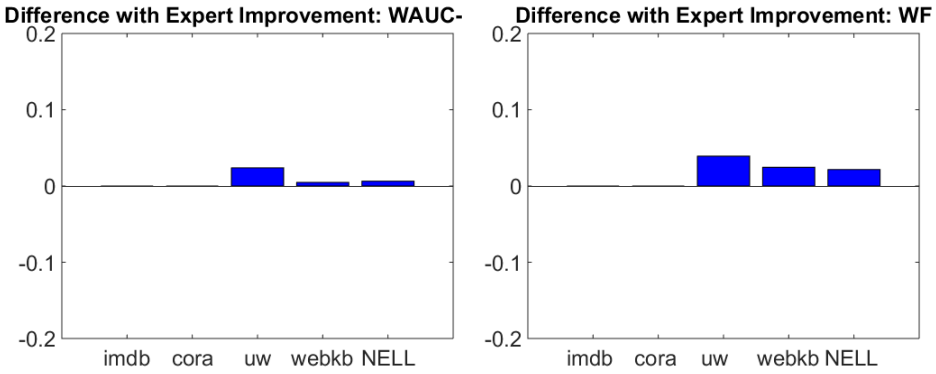


**Fig. 2.** Experimental results comparing the difference in performance when the expert improves the transferred advice. We compare both weighted-AUC ROC (**LEFT**) and weighted-F score (**RIGHT**). The more positive the result, the more impact the expert has in the IMPROVE step of LAST.

purpose of the domain is to predict which conferences take place in the same venue.

The **uw** domain [20] consists of information about universities with details like professors, courses, students, which professor teaches which course etc. The goal is to use this information and predict which professor advises a particular student. This is paired with the **webkb** domain [4] which has details of the webpage structure of universities like department webpages, course webpages, a link's source and destination page. The goal is to predict the department of a person.

**NELL** data is probabilistic data garnered through web crawling by an online machine learning system designed and deployed at Carnegie Mellon Uni-

**Table 1.** The negative to positive ratio of examples in the experimental domains.

| Domains | imdb | cora | uw | webkb | NELL |
|---|---|---|---|---|---|
| #Neg/#Pos | 14.9 | 1.7 | 548.9 | 400.5 | 7.2 |

**Table 2.** Sample source knowledge (S), target knowledge (T), and advice (A) along with its english interpretation for two of our experimental domains.

| uw | | | |
|---|---|---|---|
| S: | $LinkTo(c, a, b), Student(a), Dept(b)$ | T: | $taughtby(c1, b, q1), year(a, y1)$ |
| A1: | $< [taughtby(c1, b, q1), year(a, y1)], advisedby, \neg advisedby >$, Professors who teach courses advise students who are in some year of the program. | | |
| A2 | $< [\neg taughtby(c1, b, q1) \vee \neg year(a, y1)], \neg advisedby, advisedby >$ People who do not teach courses do not advise students in any year of the program. | | |
| NELL | | | |
| S: | $acq(comp1, comp2), sector(sect1, comp2)$ | T: | $tmplystm(tm1, tm2), plays(sport1, tm1)$ |
| A1: | $< [tmplystm(tm1, tm2), plays(sport1, tm1)], teamplyssport, \neg teamplyssport >$, Teams who play each other play the same sport. | | |
| A2: | $< [\neg tmplystm(tm1, tm2) \vee \neg plays(sport1, tm1)], \neg teamplyssport, teamplyssport >$ Teams who never play each other play different sports. | | |

versity [3]. We experiment with the sports domain here, where we predict what sport a particular team plays. The domain has details regarding members of a team, the sports an individual plays, the league in which a team plays. The source used for transfer is the finance domain, where the goal is to predict to which financial sector a company belongs.

### 5.2 Evaluation Metrics Used

Relational domains naturally suffer from extreme class-imbalance as most relationships in the world are false - most students are not advised by most professors and most actors do not work with most directors. Thus, algorithms that predict all relations as false can achieve high performance on many traditional evaluation metrics. Following previous work [22], we measure performance based on weighted-AUC ROC and weighted-F scores that weigh the high recall regions of ROC curve more than the low recall regions. Weighted-F score is defined as

$$F_\beta = (1 + \beta^2) \frac{Precision \times Recall}{\beta^2 \times Precision + Recall}$$

Note that the parameter $\beta$ controls the trade-off between the $Precision$ (percentage of correct positive predictions) and $Recall$ (percentage of positive examples correctly identified). Following [22], we use $\beta = 5$. Due to the large number of negative examples in many of the domains (as shown in Table 1), the various baselines sample from the set of negative examples. Based on previous experience, LAST samples 2 to 1 (negatives to positives) while LTL and LTL_Ref sample 5 to 1.

### 5.3 Methods Compared

We compare LAST against several baselines. To show how LAST deals with noisy data, we compare against a language-bias transfer learning method (LTL). To show how effective the human expert is in refining and improving the advice, we compare against a variant of the previous transfer learning technique that automatically refines the transferred rules (LTL_Ref). In a separate experiment, we empirically validate the expert contribution by measuring the performance gained by the IMPROVE step of LAST.

### 5.4 Results

We show our results for weighted-AUC ROC (**LEFT**) and weighted-F score (**RIGHT**) in Figure 1. In both performance measures, across all domains, our LAST algorithm performs as well as or significantly better than the baselines. More precisely, the results are significantly better than LTL and LTL_Ref except for the **imdb** domain (and **uw** for wauc-roc). As all of the methods are comparable on **imdb**, this is likely caused by this domain being easily solved (the model can be captured by a single clause). These results clearly answer (**Q1**) affirmatively, i.e., LAST is able to effectively learn with transferred advice.

As expected, the transfer learning techniques suffer from the noisy training data in several domains. In the **webkb** domain, refining the transferred knowledge (LTL_Ref) actually reduces the performance and in several other domains (**imdb**, **NELL**) there is little difference between simple transfer learning and automatically refining the rules.

The LAST algorithm has two advantages over the transfer learning baselines that allows it to effectively deal with the noisy training data. Both of these advantages directly relate to having a human-in-the-loop. Firstly, the automatic refinement (function IMPROVE from algorithm 1) allows for improvement of rules that resemble the correct knowledge, but that have been altered by the noise. Secondly, the preference labeling (function QUERY from algorithm 1) allows the expert to control the direction of the knowledge. If noise has reversed the meaning of the knowledge, the expert can effectively correct it. Figure 2 shows the difference in performance with and without the IMPROVE step of LAST. Blue bars represent the increase in performance with expert improvement. Increase in performance is seen in 3 of the 5 domains. These results show the impact of the expert, i.e., the expert is important (**Q2**).

In our experiments, we restrict the expert to manipulating a single piece of advice for each learned model. This effectively shows if we can get reasonable performance with minimal expert time while creating the system that can generate bias that mimics the expert. Therefore, the effort required by the expert to generate the advice is inherently low. It clearly follows that generating the advice from scratch would require more effort from the expert. Consequently, (**Q3**) can be answered affirmatively. In the future, we plan to quantify the value of transferred knowledge to the expert.

**Quality of the Advice:** We present sample advice for two of the domains in Table 2. Each advice includes its provenance, the rule that generated it (S), the transferred rule (T), and the final advice (A). The english interpretation of each advice rule is also provided. Note that each rule becomes two pieces of advice (and the advice is given over all possible examples in the domain). Investigating the advice in each domain, all the rules generated appear to make sense - i.e., these are rules that would possibly be naturally specified by the expert if he/she had to do without a transfer system. For example, take the advice generated for NELL which states "if a team A plays team B and team B plays a sport, then team A must also play that sport". It is clear that we are able to generate reasonable rules in all domains (**Q4**).

## 6 Conclusion

We presented a novel transfer-based human-in-the-loop framework for advice-giving in probabilistic logic models. Our goal was to develop a system that could provide human-like advice to a learning system based on transferring knowledge from a seemingly unrelated domain. The expert can then refine the knowledge generated by our LAST algorithm as needed. We demonstrated empirically the effectiveness of LAST in the presence of noisy training data. It showed the value of using transferred knowledge as "advice" instead of merely treating them as predictive rules in the target domain.

Our next step is to develop better heuristics to select the top rules that can be refined by the expert. Also, we aim to employ human guidance not only in the process of refinement of rules but also in the transfer process itself. More precisely, we aim to use humans to provide a bias during the search of the mapping between the two domains. Also, recall that this paper aims to discriminatively learn rules for predicting individual relations. Learning a generative model that can transfer across domains and provide an useful inductive bias in the target domain remains an interesting direction. Finally, extending this work to sequential decision making tasks can lead to development of human-like thinking machines.

## References


1. Bilenko, M., Mooney, R.: Adaptive duplicate detection using learnable string similarity measures. In: ACM SIGKDD (2003)
2. Blockeel, H.: Top-down induction of first order logical decision trees. AI Communications 12(1-2) (1999)
3. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E., Mitchell, T.: Toward an architecture for never-ending language learning. In: AAAI. vol. 5, p. 3 (2010)
4. Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., Slattery, S.: Learning to extract symbolic knowledge from the world wide web. In: AAAI. pp. 509–516 (1998)
5. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S.: Probabilistic inductive logic programming. Springer (2008)

6. Friedman, J.: Greedy function approximation: A gradient boosting machine. In: Annals of Statistics (2001)
7. Fung, G.M., Mangasarian, O.L., Shavlik, J.W.: Knowledge-based support vector machine classifiers. In: NIPS. pp. 521–528 (2002)
8. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning. MIT Press (2007)
9. Gros, H., Thibaut, J.P., Sander, E.: Robustness of semantic encoding effects in a transfer task for multiple-strategy arithmetic problems. In: CogSci (2015)
10. Haaren, J., Kolobov, A., Davis, J.: Todtler: Two-order-deep transfer learning. In: AAAI (2015)
11. Jones, W., Moss, J.: Interruption-recovery training transfers to novel tasks. In: CogSci (2015)
12. Kumaraswamy, R., Odom, P., Kersting, K., Leake, D., Natarajan, S.: Transfer learning via relational type matching. In: ICDM (2015)
13. Kunapuli, G., Odom, P., Shavlik, J.W., Natarajan, S.: Guiding autonomous agents to better behaviors through human advice. In: ICDM. pp. 409–418 (2013)
14. Mihalkova, L., Huynh, T., Mooney, R.: Mapping and revising markov logic networks for transfer learning. In: AAAI. vol. 7, pp. 608–614 (2007)
15. Mihalkova, L., Mooney, R.: Bottom-up learning of markov logic network structure. In: ICML. pp. 625–632 (2007)
16. Natarajan, S., Kersting, K., Khot, T., Shavlik, J.: Boosted Statistical Relational Learners: From Benchmarks to Data-Driven Medicine. Springer (2015)
17. Natarajan, S., Khot, T., Kersting, K., Gutmann, B., Shavlik, J.: Gradient-based boosting for statistical relational learning: The relational dependency network case. Machine Learning 86, 25–56 (2012)
18. Odom, P., Khot, T., Porter, R., Natarajan, S.: Knowledge-based probabilistic logic learning. In: AAAI (2015)
19. Pan, S., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359 (Oct 2010)
20. Richardson, M., Domingos, P.: Markov logic networks. Machine learning 62(1-2), 107–136 (2006)
21. Towell, G.G., Shavlik, J.W.: Knowledge-based artificial neural networks. Artif. Intell. (1994)
22. Yang, S., Khot, T., Kersting, K., Kunapuli, G., Hauser, K., Natarajan, S.: Learning from imbalanced data in relational domains: A soft margin approach. In: ICDM (2014)