# Actively Interacting with Experts: A Probabilistic Logic Approach

Phillip Odom (✉) and Sriraam Natarajan

Indiana University,
Bloomington, Indiana
{phodom,natarasr}@indiana.edu

**Abstract.** Machine learning approaches that utilize human experts combine domain experience with data to generate novel knowledge. Unfortunately, most methods either provide only a limited form of communication with the human expert and/or are overly reliant on the human expert to specify their knowledge upfront. Thus, the expert is unable to understand what the system could learn without their involvement. Allowing the learning algorithm to query the human expert in the most useful areas of the feature space takes full advantage of the data as well as the expert. We introduce *active advice-seeking* for relational domains. Relational logic allows for compact, but expressive interaction between the human expert and the learning algorithm. We demonstrate our algorithm empirically on several standard relational datasets.

## 1 Introduction

Probabilistic logic models (PLMs) [8, 3] combine the expressive power of first-order logic and the ability of probability theory to model noise and uncertainty. They have been inspired by databases [6, 9] and by logic [4, 5]. Given their expressivity, several powerful learning algorithms have been developed that allow for learning from interpretations [5, 18] and learning from entailment [23, 4]. While efficient algorithms have been developed to learn the parameters of these models (either weights or probabilities), full model-learning (also called *structure learning* to denote learning of the logical structure) remains a challenging task. Recently, methods based on ensemble learning have been proposed that allow for efficient structure learning for PLMs [16].

These methods essentially rely only on data. Given that the primary assumption is that data can be noisy, restricting humans to be mere labelers of the data, as is done in many popular approaches, is inefficient. Recently, a formulation for incorporating prior knowledge as *preferences* over labels for the ensemble learning method was proposed [19]. The key idea was to explicitly trade-off between the label preferences suggested by the human expert and the posterior label distributions obtained from the data. It was demonstrated that advice was particularly useful where there was targeted noise. For example, missing certain regions in a segmentation task, or missing stop signs when creating driving demonstrations.

While the framework of Odom et al. [19] does not merely treat the given advice as "prior" knowledge, it assumes that all the advice is provided up-front before the learning takes place. Not only is this a potentially time consuming task for the experts, but it is also highly likely that they, not being experts in machine learning or probabilistic logic, would find it difficult to identify the domain knowledge that might be optimal for the learning algorithm. Hence, inspired by active learning [24], we propose *active advice-seeking* that aims to determine the regions of (relational/logical) feature space that is ideal for obtaining advice. For instance, will the accuracy of a model learned to predict heart attacks be higher if advice is given about the population who is overweight and has high blood pressure or about the population which smokes but exercises regularly? The answer is not clear but this is where active advice-seeking should be helpful. The goal of active advice-seeking is to lessen the responsibility of the expert both in terms of the effort that must be spent in specifying the advice, as well as the necessity that the expert understands the intricacies of the algorithm. The algorithm will automatically identify the regions of the feature space where the advice will be useful.

More precisely, the proposed algorithm presents a set of conjunctions of predicates as queries to the expert. The size of the set is pre-determined by a budget given by the expert (i.e., the algorithm and the expert agree in advance for the number of allowable queries). In order to compute the clause that should be queried, the algorithm learns a model from only the data to compute a score for each example, then it uses a regression clause learner to fit the scores. The best clause is presented to the expert who provides a preference over the labels. For instance, in a university domain, the clause could be of the form *prof(X)* $\wedge$ *student(Y)* $\wedge$ *paper(P,X)* $\wedge$ *paper (P,Y)*. The expert could then prefer the label to be $advisedBy(X, Y)$. Essentially the system is asking the expert, what is your choice of label if a student and a professor are co-authors? The expert replies saying, I prefer the student to be advised by the professor. Note that this is a "soft" preference in that this preference may not always hold. This preference is then explicitly weighed against the data while learning the model.

We make the following key contributions: first, we introduce the notion of advice-seeking to the probabilistic logic model (PLM) community (and the general AI community). Second, we adapt a recent successful knowledge-based probabilistic logic learning algorithm to seek advice from the human expert. Third, we present the first relational algorithm that can go beyond data and interactively solicit input from the expert. Finally, we demonstrate using experiments, that such an approach is robust in learning from noisy data.

The rest of the paper is organized as follows: we first introduce the required background on PLMs and active learning. Then, we present our learning approach before presenting empirical evaluations. Finally, we conclude the paper by outlining areas for future research.

## 2 Background

Techniques for incorporating expert knowledge into learning are a key precondition for any active advice-seeking approach to be successful. We aim to introduce a broad learning paradigm that can use any method that incorporates prior knowledge. To that effect, we cover one advice-based framework which we will use to empirically validate our approach.

### 2.1 Advice-based PLMs

While there have been many knowledge-based systems developed for propositional models [27, 7, 26, 14, 11], work on probabilistic logic models (PLMs) has not progressed as far. In PLMs, the expert is typically used to define some prior structure that can either be used as the complete structure or locally refined.

Recently, Odom et al. [19] introduced a knowledge-based PLM method that learns seamlessly from data and any expert knowledge. While making use of Relational Functional Gradient-Boosting (RFGB) to learn the structure and parameters of the model simultaneously [17], they incorporate expert preferences which guide the structure and parameters to more robust models.

Extending previous work that considered knowledge as propositional Horn clauses [27, 7, 12], they considered their advice as first-order logic Horn clauses. Thereby, allowing experts to give advice over different granularities of examples. The body of the clause specifies the examples over which the expert would like to give advice, while the head of the clause gives the preferred and avoided labels. For example, a cardiologist might suggest that patients whose close relatives had heart problems are more likely to have a heart problem.

Odom et al. [19] incorporate this expert knowledge into RFGB [17] which learns a series of relational regression trees [2]. These relational regression trees have first-order logic literals in the nodes and regression values at the leaves. Functional gradient-boosting aims to capture the error in the current model in a regression tree and then adds this regression tree to the model. The final model is a sum over all of the learned trees.

The gradients used by Odom et al. [19] incorporate an additional term in the optimization function that pushes the model in the direction of the expert advice (represented by $n_t$ and $n_f$, the number of advice which say that example $x_i$ should be preferred/avoided)[1]

$$\Delta(x_i) = \alpha \cdot (I(y_i) - P(y_i;\psi)) + (1-\alpha) \cdot [n_t(x_i) - n_f(x_i)]$$

While this approach has shown positive results in several difficult tasks, it still requires the expert to specify *all of the advice in advance*. Given a particular dataset, deciding the most useful advice is not a trivial problem. This problem is exacerbated by the fact that the expert could potentially have no expertise in machine learning. *Active advice-seeking* aims to alleviate this issue by querying

[1] Note the difference to standard (only data) RFGB which optimizes $(I(y_i) - P(y_i;\psi))$.

the expert directly, using the training data as a guide to select the most useful queries. Previous work on *active advice-seeking* is limited to propositional queries in sequential decision making problems [20]. Grouping ground states into queries allowed the proposition algorithm to maximize the impact of the human expert. However, lifting advice to be relational as we do in this work is a more powerful and principled approach.

### 2.2 Active Learning

Active Learning is a related research problem where the goal is to make use of an expert that can provide the labels of examples [24]. Pool-based active learning approaches assume a pool of unlabeled examples from which the learning algorithm should choose. In *active advice-seeking*, this pool of examples is the training set. While there are labels in the training set, it is assumed that either there is not sufficient training data (and thus there is missing knowledge) or the training data is noisy and so the labels should not be fully trusted. So while active learning aims for finding the labels of the examples, we are soliciting advice.

Most active learning methods repeat the following general steps:

1. Learn a model from training data
2. Compute uncertainty over unlabeled data
3. Select examples based on uncertainty and solicit label
4. Add labeled examples to training set

The process begins by learning a model with the current set of labeled data. This model is then used to compute some measure of uncertainty (this could be entropy, KL-divergence or other measures) that suggests how likely the model would correctly predict the unlabeled examples. Consider a simple, linear classifier with two possible unlabeled examples, one located close to the decision boundary with the other located far from the boundary. The example close to the decision boundary is more likely to effect the decision boundary and would be selected for labeling.

This cycle accumulates the best examples to label at each step and has been shown to be effective especially in domains where there is a dearth of data available. However, labeling individual examples is not an effective use of human experts availability. Allowing expert's to give advice results in the expert being able to select the ideal granularity of advice (over a single example or many examples). *Active advice-seeking* aims to effectively use human experts by providing clauses instead of ground examples. Not only does this allow for automatically selecting the granularity of advice, but it also provides a compact description of the most uncertain examples.

A particular active learning paradigm that is closer to our work is the work of Rashidi and Cook [21]. In their work, they cluster informative examples and run a rule induction algorithm (such as C4.5) to generate a rule based query to which the expert can provide a label. The similarity to our approach lies in the use of a rule to ask the query. The two key differences are that, first, ours is a relational learning algorithm that goes beyond flat feature vectors. Second, the rule was used to obtain a label that was used for all the examples that satisfy that rule. In our case, we go beyond labels and solicit human advice as preferences over logical rules.

Active learning has been considered for relational data particularly, with the focus of querying for node labels based on the structure of the network [13, 1, 22, 15] which have been studied under the broad area of active inference in relational domains. Particularly relevant to our paper are three of the most recent works - ALFNET [1], the RAL algorithm [13] and FLIP [25]. ALFNET employed uncertainty sampling to generate committee-based network clusters (which consisted of three classifiers) in order to query the expert. A related work in this direction is the RAL algorithm that used a utility metric with network variance as the criteria. This variance was used since the RAL algorithm is interested in across-network classifications. While we do not employ this heuristic, our algorithm can handle across-network classifications due to the underlying logic-based ensemble learner. Finally, the FLIP algorithm by Saha et al, extends the notion of active inference by considering several query selection methods and evaluates them on single and multi-labeled networks. Our algorithm is similar in spirit to ALFNET in that we employ uncertainty sampling as well but our query is generated using clauses learned through logic programming. An important difference to the RAL, FLIP and ALFNET algorithms is that we query for preferences over the relations instead of the actual labels.

## 3 Relational Active Advice-Seeking

The aim of relational active advice-seeking is to offload the task of selecting areas of the feature space to give targeted advice from the human expert to the learning algorithm. In relational models, experts are often asked to define the logical structure of the model with the parameters learned from data. However, it is important to be able to learn the full model (structure and parameters) especially in complex, real world domains. Experts can still provide valuable input about targeted areas of the feature space. The wide variety of potential expert advice complicates the advice-giving process and can lead the expert to give correct, but not relevant advice.

Previous work on advice-giving requires significant effort on the part of the expert to determine the relevant advice [19, 12]. If the expert provides exhaustive advice, the learning algorithm will be able to learn an accurate classifier. However, the experts time is often limited and only a few queries can be answered. These queries should not be redundant, focusing on areas that are well covered by the data. Instead, they should focus on areas where the learning algorithm cannot distinguish the correct label or behavior. Thus, we extend relational advice-taking methods to active advice-seeking. Each part of our formulation is shown in Figure 1. It consist of the active advice-seeking component that is capable of generating queries and interacting with the human expert as well as the knowledge-based learning algorithm which learns from the expert provided knowledge and any available training data.

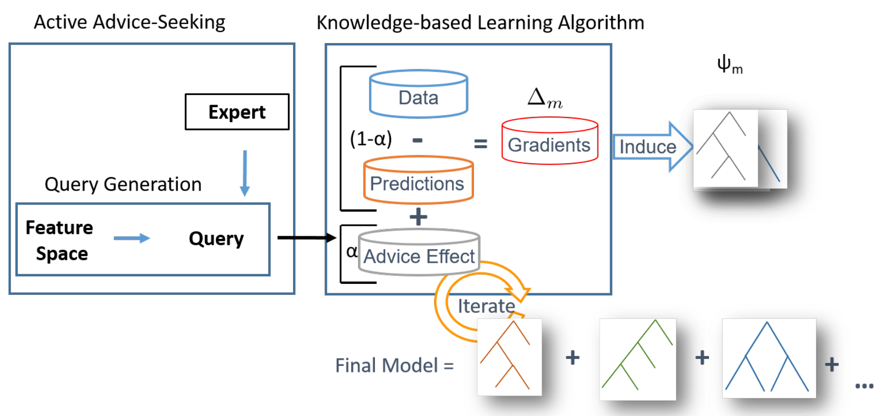


**Fig. 1.** An overview of our framework for actively interacting with human experts. The learner is responsible for selecting where to query the expert.

## 4 Problem Formulation

The overall goal of our algorithm is to identify regions of the feature space that the agent is most uncertain about and query the expert for advice on these regions. In the propositional case, this was handled by simply clustering examples based on the distribution over the labels and querying the expert over this cluster [20]. However, this heuristic may not suffice for relational tasks since there are typically more negative examples than there are positives. Fortunately, the use of a rich representation such as first-order logic naturally allows us to query over the most uncertain regions of the feature space.

We represent the regions of feature space as conjunctions of predicates. Intuitively, this corresponds to grouping examples such that a particular condition is satisfied. More precisely, the goal of our algorithm is to select a set of conjunctions of first-order logic atoms about which to query the expert. These queries concisely describe the set of training examples to which the advice will apply. In order to select relevant areas of the feature space, the algorithm learns a clause (model) based on scores of the given examples. The goal of this learned model is to group similar examples based on their assigned score which measures the importance of that query. Queries have low scores if the algorithm is confident in its prediction, Otherwise, the query will receive a high score, making it more likely to be selected by the active advice-seeking algorithm. We explain the clause generation later in this section. We will now formally define advice:

**Definition 1** *A set of advice (A) is defined as a series of relational queries ($Q_i$) and the experts corresponding response ($R_i$), ie. ($A =< (Q_1, R_1), (Q_2, R_2), ..., (Q_n, R_n) >$).*

The algorithm solicits a sequence of queries that depend on the scoring function that will be discussed in detail later. The number of queries is dependent on the difficulty of the problem and the availability (query budget) of the expert.

**Definition 2** *A Relational Query (Q) is defined as a conjunction of literals ($\wedge f_i$), which defines the set of examples to which the advice will be applied. Q will be shown to the human expert.*

**Definition 3** *An Expert Response (R) is defined as a set of preferred labels ($l+$), and a set of avoided label ($l-$) given with respect to a relational query. Note that both $l+/l-$ could be empty if the expert does not understand Q or if the query does not separate different classes.*

If the expert is not satisfied with the query - possibly because the query does not properly delineate between labels - then the expert can provide no preferred or avoided labels. Such a query is not useful to the learning algorithm and squanders the time of the expert. The relational query and its accompanied response represent a single piece of advice that can be utilized by the knowledge-based learning algorithm. We now present an illustrative example before discussing the algorithm in detail.

### 4.1 Illustrative Example

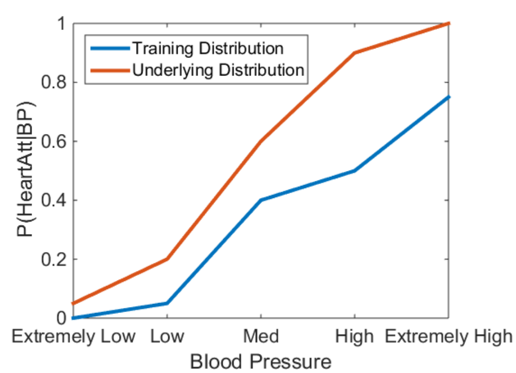


**Fig. 2.** Example showing the distribution of heart attacks given blood pressure for an observed and underlying distribution. The difference in these distributions could cause an expert to give advice that is not customized with respect to the training distribution.

Consider the example of heart attack prediction given clinical information about the patients such as their blood pressure. The training set (e.g. one particular county in Wisconsin) might show all patients having a lower risk of heart attack, with patients having high blood pressure having an especially low incidence of heart attacks. This systematic difference could be attributed to local factors. The local county data (the training set) could be shown in Figure 2 in blue, while the true distribution for the entire nation could be shown in red.

Now consider soliciting advice about heart attacks and blood pressure from a cardiologist in California. Being unfamiliar with Wisconsin, the cardiologist might give broad, straight-forward advice. However, such knowledge might already follow from the training data. Examples of such advice include "extremely high blood pressure leads to heart attacks" and "heart attacks are not likely with low blood pressure". While these pieces of advice are valid, they are not the most relevant advice for this particular learning problem.

If the algorithm had the ability to solicit advice, then it could direct the expert to give the most relevant advice at any point. Our proposed algorithm will identify areas in the data that are unclear and will instead query the expert automatically with "How likely are heart attacks when the blood pressure is high, but not extreme". This is likely the most useful advice given the data. This approach not only benefits the learning algorithm, but reduces the burden on the expert who is only required to answer specific questions.

**Algorithm 1** Actively Seeking Advice for PLMs (ASAPlm)

**function** ASAPLM($D$,$E$,$MaxQuery$)
  $A = \emptyset$
  $M$=RFGB($D$) ▷ Model from Noisy Data
  **for** $x_i \in D$ **do** ▷ Compute Uncertainty per Example
    $R(x_i) = H(x_i)$
  **end for**
  $AQ$=LRC($D, R$) ▷ Learn Regression Clauses
  **for** $i = 1$ to $MaxQuery$ **do** ▷ Query Expert
    $AQ_q$ =MAXSCORE($AQ$)
    $AQ = AQ - AQ_q$
    $< AQ_q, R >$=QUERY($E, AQ_q$)
    $A = A \cup < AQ_q, R >$
  **end for**
  $M_F$=ADVLEARNER($A, D$) ▷ Learn with Advice
  **return** $M_F$
**end function**

### 4.2 The Algorithm

Our proposed approach involves generating a set of queries, scoring those queries to rank them according to their usefulness, and finally soliciting the most useful queries to the human expert. The number of queries that can be requested depends on the problem (more difficult domains require more knowledge) and the availability of the human expert. The complete active advice-seeking algorithm (ASAPlm) is shown in Algorithm 1. We will address each of these vital components in turn.

**Generating and Scoring Queries** Recall that in standard active learning, a model is learned from labeled data and using this model, some uncertainty measure is calculated to identify the most uncertain unlabeled example to query the expert. We take a similar approach with an important change. We learn an ensemble of relational regression trees using RFGB on the noisy data (line 3 of the algorithm) and compute the entropy over the examples given this model (lines 4-6). Following active learning, we define the score of an example as the entropy of the model's prediction (line 5 of Algorithm 1), ie,

$$H(x_i) = \sum_{l \in Labels} P_l(y_i|x_i) log(P_l(y_i|x_i))$$

where $P(y_i|x_i)$ is learned using RFGB. Such uncertainty measures have performed extremely well in many active learning methods and similar results can be shown over relational data. The key difference is that the uncertainty is based on all of the training examples that satisfy the query. In our empirical evaluation, we focus on entropy as our uncertainty measure. However, the framework is broad and allows for the selection of the most appropriate uncertainty function for the problem at hand.

Then these scores are used as regression values for the corresponding example and a set of weighted first-order-logic clauses are learned that can potentially group these examples (line 7, function LRC). We learn relational regression trees using RFGB as our implementation. These clauses are presented to the expert according to the learned weights. We learn these weighted clauses through an adaptation of RFGB where instead of learning $P(y_i|x_i)$, we want to learn a model for the uncertainty values of $x_i$ (by fitting regression trees). The key intuition is that the regression trees find clauses that apply to examples with similar uncertainties. Note that unlike in discriminative learning where there are positive and negative examples, regression does not treat positive and negative examples differently. Every example has a uncertainty value and regression is just trying to fit those values. The learned clauses represent a set of possible queries from which the algorithm can select.

**Querying the Expert** After the queries have been generated and ranked, they can be used to solicit advice from the human expert. For a given relational query, the expert should supply the suggested preferred labels (should be considered more likely) and the avoided labels (should be considered less likely). Alternatively, the expert could decline to answer if the query is too general or incomprehensible. Declining is an indication that the active advice-seeking algorithm is not selecting appropriate queries.

**Advice-based Learner** Given the advice, the final step is to utilize the advice-based learner to learn from both the training data as well as the expert advice. An ideal algorithm should trade-off between the sources of knowledge when they offer contradictory information. For the purposes of empirical validation, we utilize KBPLL [19] as our advice-based learner. It combines the target distribution of the training data and the distribution suggested by the advice to find a robust model (refer to section 2).

Overall, the proposed approach to active advice-seeking aims to effectively utilize the human expert by generating queries. These queries are targeted based on the perceived weaknesses in the training data. We now thoroughly investigate the active advice-seeking algorithm.

## 5 Experiments

Through our experiments, we aim to answer the following questions:

**Q1**: Does active advice-seeking result in more effective learning?
**Q2**: Is our algorithm robust to both random and systematic noise?
**Q3**: Is advice an efficient form of communication between algorithm and expert?

### 5.1 METHODS

We compare our method against two baselines. To evaluate our query generation method, we compare against learning with randomly generated queries (*Random Queries*). Note that the expert still gives the correct answer for the particular query generated. To evaluate the effectiveness of active advice-seeking, we compare against learning with no advice (*No Advice*). This represents the effectiveness of traditional machine learning systems that do not make use of expert knowledge. We also discuss the quality of the advice that is generated in each domain. Given our experience with the domains, we take the role of expert to answer the queries.

In all the experiments, we compare the accuracy of learned model. To show that our algorithm is capable of correcting noisy data, we added noise equal to 25% of the positive examples. Note that in the relational space the number of negative examples typically greatly outnumbers the positive examples. This means that the impact of the noise is much less than 25%. To show that our algorithm is capable of correcting systematic noise, we label examples incorrectly in a targeted region of the feature space. The synthetic heart attack dataset and driving domain are domains where systematic noise is natural. Heart problems effect different regions or ethnic groups in different ways and many drivers consistently drive over the speed limit and roll through stop signs. For the remaining datasets, imdb, webkb and uw, we have experimented with both systematic and noisy data. Each randomly noisy experimental domain has either 4 or 5 folds and we randomly add noise 5 times for each fold. Each systematically noisy experiment generated data for each fold or was repeated 5 times. For our relational advice-based learning algorithm, we use KBPLL [19] with $\alpha = 0.25$.

| Domain | Prediction Task (Possible Labels) | Type of Noise |
|---|---|---|
| Driving | moveLeft,moveRight,stayInLane | Systematic |
| Synthetic | heartAttack | Systematic |
| IMDB | workedUnder | Systematic/Noisy |
| WEBKB | faculty | Systematic/Noisy |
| UW | advisedBy | Systematic/Noisy |

**Table 1.** Describes the prediction task of each of the experimental domains as well as the kind of noise used in the experiments.

### 5.2 DOMAINS

We have a variety of standard relational datasets as well as an imitation learning dataset focused on driving. An overview of each domain and the corresponding typed of noise (the datasets are either systematically noisy or randomly noisy) used in conjunction with that domain is shown in Table 1.

**IMDB** : This dataset is a movie database that consists of movies, actors, directors and their various genres. Our goal is to predict the workedunder relationship (ie which actors worked on movies under a particular director). This dataset consist of 5 folds.

**WEBKB** : This dataset is a university dataset that consists of webpages and their hyperlinks. Our goal in this domain is to predict which webpage belongs to a faculty member based on the webpages and their linking structure. This dataset has 5 folds.

**UW** : This dataset is a university dataset that consists of professors, students, courses, and publications each having various relationships and features. Our goal is to predict the advisedby relationship. This dataset has 4 folds.

**SYNTHETIC** : The goal of the synthetic dataset, from the illustrative example, is to predict heart attacks given the blood pressure. There is a systematic difference (see Figure 2) between the training set and the testing set. This dataset was generated 5 independent times.

**DRIVING** : The driving domain focuses on navigating down a 5-lane highway, avoiding the other cars on the road [10]. The possible actions are to stay in the current lane or change lanes to the left or right. The size of the training set and testing set are 100 trajectories consisting of 10000 total training examples.

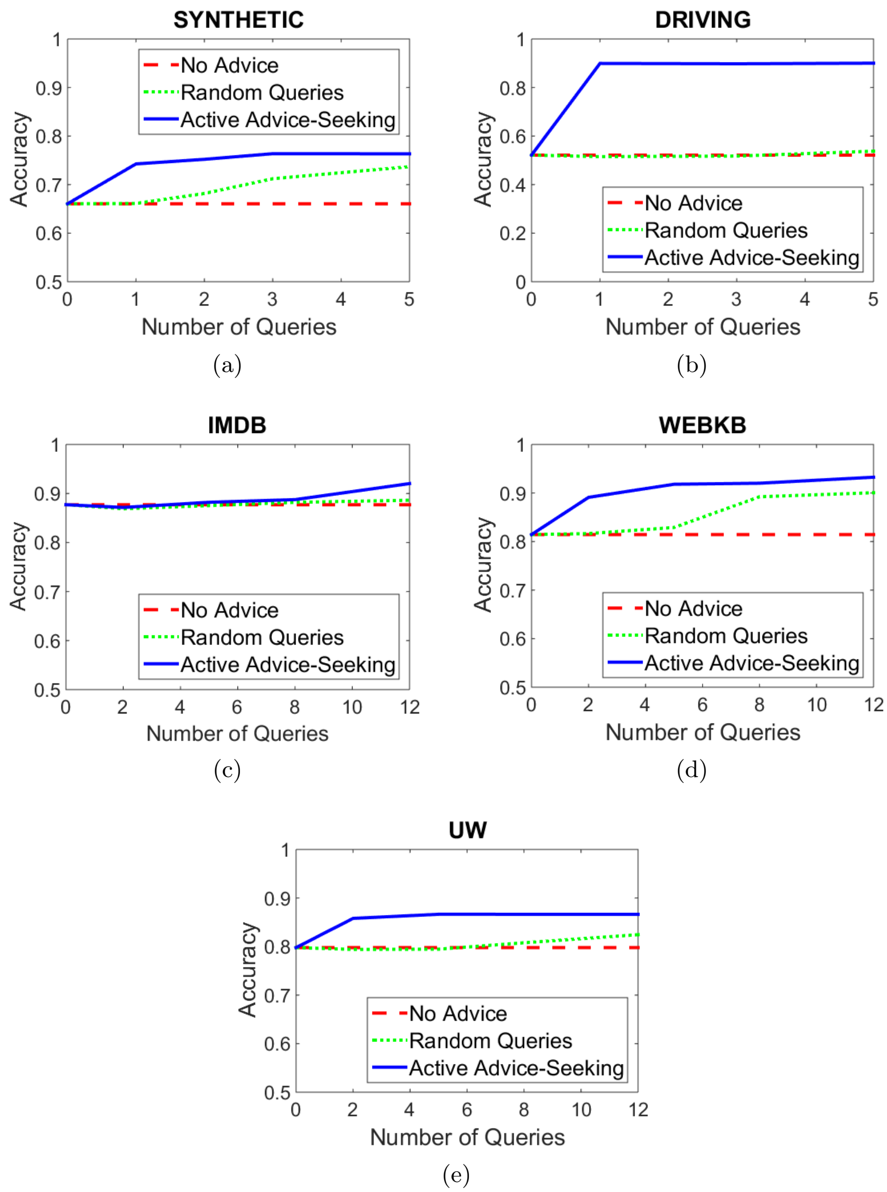


**Fig. 3.** The learning curves for the experiments with systematic noise. Each learning curve shows accuracy as the number of queries to the expert increases. We compare Active Advice-Seeking to Random Queries and No Advice.

### 5.3 Systematic Noise

The results with systematic noise (Figure 3) are shown for the synthetic and driving domains as well as each of the standard relational datasets. Together they

| Domain | Query Generated |
|---|---|
| Driving | What if there is a car in the left lane? |
| Synthetic | What if a person has medium to high blood pressure? |
| IMDB | Do female actors work under people in crime movies? |
| WEBKB | What is the title of students working on projects? |
| UW | What is the relationship between students and TA's? |

**Table 2.** The top queries generated in each domain for the systematically noisy datasets. Experts respond to these queries by providing $l+/l-$ from Table 1.

show the power of our proposed approach when dealing with systematic noise. In most datasets, the algorithm is capable of selecting useful queries immediately, providing significant impact. Random queries demonstrate gradual performance gains in the synthetic and webkb domains, but fail to have a positive effect on the other domains. While random queries do not cause performance to degrade, they have an extremely difficult time isolating systematic noise especially when there are more features. A key reason there is very little change in these domains is that the queries generated were ambiguous and useful only for a few examples. For instance, a common query in the driving domain is "What action should I take if there is a car both to my left, right, AND in front". While this is a possible scenario, it is not likely in this dataset and there is no obvious advice to give for these states. Alternatively, the queries generated from the active advice-seeking algorithm select more relevant and overall useful queries. Thus, **Q1** is answered affirmatively in that our proposed approach is able to learn effectively in the presence of systematic noise.

### 5.4 Random Noise

The standard relational domains (Figure 4) are used to show that even when noise is random, our proposed method can still generate high-quality queries to the expert. Random noise should be more difficult for our algorithm, as there may not be specific regions of the feature space that need attention. However, across all three domains, our proposed approach achieves consistent success, generating performance gains with each query. In contrast, randomly generated queries can yield positive performance (as in imdb or uw), or actually result in a model that is worse than relying on the data (as in webkb). It may seem counter intuitive for advice to be harmful. However, consider the query "Is a student advised by a professor". While it may seem that the advice should be that students are advised by professors, there are many student and many professors. Therefore, such an advice could result in many false positives as a student is not advised by most professors. Thus, our proposed approach is robust to random noise as well as systematic noise (**Q2**).

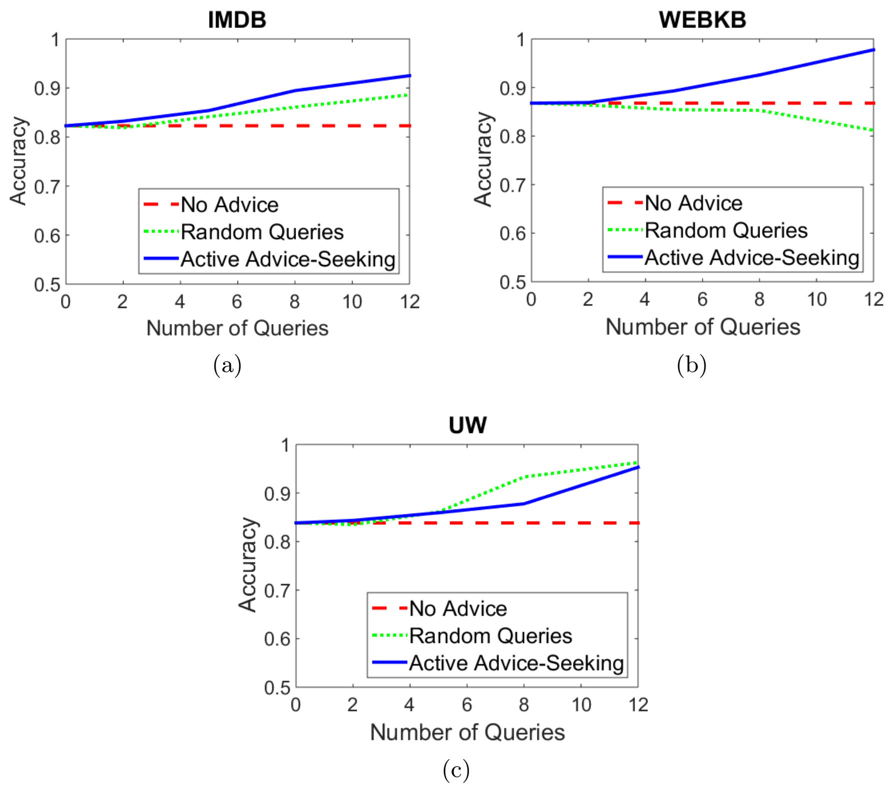


**Fig. 4.** The learning curves for the experiments with random noise. Each learning curve shows accuracy as the number of queries to the expert increases. We compare Active Advice Seeking to Random Queries and No Advice. As previously, randomness (for Random Queries) does not come from incorrect answer by the experts, but rather from randomly generated queries.

### 5.5 Quality of Advice

The preceding empirical results show that our proposed approach is able to generate relevant queries that yield significantly higher accuracy in nearly all of the domains for both systematic and noisy experiments. However, the interpretability of the queries is vital as the experts need to easily comprehend the queries in order to give the proper advice. Table 2 shows the top query generated for each domain (systematic noise). In the driving domain, the query asks what action to take when there is a car in the left lane. The expert response would be to stay in the current lane. As another example, in the uw domain, the query asks about the relationship between students and TAs. While TAs might help teach students, the advice would say that TAs cannot advise students. The best queries are heavily influenced by the noise in the training set. Overall, the queries are concise (as shown in Table 2) and effective (as shown in the empirical validation). Thus, advice is an efficient form of communication (**Q3**).

## 6 Conclusion

We presented the first advice seeking framework for PLMs. Our method, inspired by active learning, queries the expert with sub-spaces of the feature space where advice can be provided as preferences over labels. The key insight is that the learning algorithm can better query the expert based on the uncertainty in the data as compared to the expert providing all advice pieces in advance. Our experimental results across standard data sets proved that such a method is indeed effective in soliciting useful advice. It must be mentioned our work is inspired by and bridges three promising areas of research inside machine learning - knowledge elicitation, active learning and PLMs. It extends knowledge elicitation to PLMs for the first time. It builds upon the success of active learning in relational tasks by soliciting advice (as preferences) instead of simple labels as done in previous research. Finally, it contributes to PLMs by making the learning algorithm go beyond merely using data by providing a natural way of interacting with the human expert.

Evaluating on larger data sets such as electronic health records is an important future direction. EHRs in particular can provide the opportunity to interact with domain experts who could provide advice potentially as qualitative statements - increase in one risk factor can increase the risk of a disease. Another interesting direction is exploring the different measures of uncertainty for grouping the different examples. A third direction could be to consider more types of advice that have been previously employed in machine learning. Learning from multiple experts by weighing them explicitly is another direction that we will explore. Finally, performing user studies on more sophisticated test beds is an interesting research direction.

### Acknowledgments

The authors thank the Army Research Office (ARO) grant number W911NF-13-1-0432 under the Young Investigator Program.

## References

1. Bilgic, M., Mihalkova, L., Getoor, L.: Active learning for networked data. In: ICML (2010)
2. Blockeel, H.: Top-down induction of first order logical decision trees. AI Communications 12(1-2) (1999)
3. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S.: Probabilistic inductive logic programming. Springer (2008)
4. De Raedt, L., Kimmig, A., Toivonen, H.: Problog: A probabilistic prolog and tis application in link discovery. In: IJCAI (2007)

5. Domingos, P., Lowd, D.: Markov Logic:An Interface Layer for Artificial Intelligence. Morgan & Claypool (2009)
6. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: IJCAI (1999)
7. Fung, G., Mangasarian, O.L., Shavlik, J.W.: Knowledge-Based support vector machine classifiers. In: NIPS. pp. 01–09 (2002)
8. Getoor, L., Taskar, B.: Introduction to statistical relational learning. Cambridge: MIT Press (2007)
9. Heckerman, D., Meek, C., Koller, D.: Probabilistic entity-relationship models, prms, and plate models. In: ICML (2004)
10. Judah, K., Fern, A., Tadepalli, P., Goetschalckx, R.: Imitation learning with demonstrations and shaping rewards. In: AAAI (2014)
11. Kunapuli, G., Bennett, K.P., Shabbeer, A., Maclin, R., Shavlik, J.W.: Online knowledge-based support vector machines. In: ECML. pp. 145–161 (2010)
12. Kunapuli, G., Odom, P., Shavlik, J., Natarajan, S.: Guiding autonomous agents to better behaviors through human advice. In: ICDM (2013)
13. Kuwadekar, A., Neville, J.: Relational active learning for joint collective classification models. In: ICML (2011)
14. Le, Q.V., Smola, A.J., Gärtner, T.: Simpler knowledge-based support vector machines. In: ICML. pp. 521–528 (2006)
15. Macskassy, S.: Using graph-based metrics with empirical risk minimization to speed up active learning on networked data. In: KDD (2009)
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(1) (2012)
18. Natarajan, S., Tadepalli, P., Dietterich, T., Fern, A.: Learning first-order probabilistic models with combining rules. Annals of Mathmatics and AI 54(1) (2008)
19. Odom, P., Khot, T., Porter, R., Natarajan, S.: Knowledge-based probabilistic logic learning. In: AAAI (2015)
20. Odom, P., Natarajan, S.: Active advice seeking for inverse reinforcement learning. In: AAMAS (2016)
21. Rashidi, P., Cook, D.: Ask me better questions: Active learning queries based on rule induction. In: KDD (2011)
22. Rattigan, M., Maier, M., Jensen, D.: Exploiting network structure for active inference in collective classification. In: IDM (2007)
23. Sato, T., Kameya, Y.: Prism: A symbolic statistical modeling language. In: IJCAI (1997)
24. Settles, B.: Active Learning. Morgan & Claypool (2012)
25. Tanwistha Saha, Huzefa Rangwala, C.D.: Flip: Active learning for relational network classification. In: ECML (2014)
26. Torrey, L., Walker, T., Shavlik, J., Maclin, R.: Using advice to transfer knowledge acquired in one reinforcement learning task to another. In: ECML (2005)
27. Towell, G., Shavlik, J.: Knowledge-based artificial neural networks. Artificial Intelligence 69, 119–165 (1994)