User Friendly Automatic Construction of Background Knowledge: Mode Construction from ER Diagrams

Alexander L. Hayes
University of Texas at Dallas
Alexander.Hayes@utdallas.edu

Mayukh Das
University of Texas at Dallas
Mayukh.Das1@utdallas.edu

Phillip Odom
Indiana University, Bloomington
phodom@indiana.edu

Sriraam Natarajan
University of Texas at Dallas
Indiana University. Bloomington
Sriraam.Natarajan@utdallas.edu

ABSTRACT

One of the key advantages of Inductive Logic Programming systems is the ability of the domain experts to provide background knowledge as modes that allow for efficient search through the space of hypotheses. However, there is an inherent assumption that this expert should also be an ILP expert to provide effective *modes*. We relax this assumption by designing a graphical user interface that allows the domain expert to interact with the system using Entity Relationship diagrams. These interactions are used to construct modes for the learning system. We evaluate our algorithm on a probabilistic logic learning system where we demonstrate that the user is able to construct effective background knowledge on par with the expert-encoded knowledge on five data sets.

KEYWORDS

Feature selection, Logical and relational learning, Entity relationship models, Interaction paradigms

1 INTRODUCTION

Recently, there has been an increase in the development of algorithms and models that combine the expressiveness of first-order logic with the ability of probability theory to model uncertainty. Collectively called Probabilistic Logic Models (PLMs) or Statistical Relational Learning models (SRL) [7, 19], these methods have become popular for learning in the presence of multi-relational noisy data. While effective, learning these models remains a computationally intensive task. This is due to the fact that the learner should search for hypotheses at multiple levels of abstraction.

Consequently, methods whose search strategies are inspired from Inductive Logic Programming (ILP) have been introduced to make learning more efficient [13, 15]. These methods have demonstrated arguably some of the best results in several benchmark and real data sets. While effective, the key issue with these methods is that they require the domain expert to also be an expert in ILP—thus providing the right set of directives for learning the target concepts. These additional directives, typically called *modes*, restrict the search space such that the learning of these probabilistic clauses is efficient. Many real users of these systems, especially those who fail to learn good models with these algorithms, may not able to select the correct modes to guide the search. The consequence is that many of the learning procedures get stuck in a local minimum or get timed out resulting in sub-optimal models.

One way that this problem has been addressed in literature is by employing databases underneath the learner to improve the search speed [11, 16, 23]. While these systems have certainly improved the search, recent work by Malec et al. [11] clearly demonstrated the need for modes to achieve effective learning even when using databases. Their work showed an order of magnitude improvement over the standard state-of-the-art PLM learning system. However, their work also required the modes to be specified for attaining this efficiency.

Inspired by their success, we propose a method for specifying modes from a database perspective. Specifically, we propose to employ the use of Entity Relationship (ER) diagrams as the graphical tools based on which an user could specify modes. The key intuition is that the modes specify how the search is conducted through the space of hypotheses. When viewed from a relational perspective, this can be seen as specifying the parts of the relational graph that are relevant to the target concept. We provide an interface that allows for an user to guide the PLM learners using ER diagrams. Our interface automatically converts the user inputs on ER diagrams to mode definitions that are then later employed to guide the search. Our work is also inspired by the work of Walker et al. [21] where a UI was designed to provide *advice* for a PLM learner. While their UI was domain-specific, our contribution is a generalized approach to utilize any ER diagram to automatically construct background knowledge for logic-based learners. Our work is also related to the other work of Walker et al. [22] where the mode construction was automated using a layered approach which relied on successively broadening the search space until a relevant model was found. While their work was effective, due to the layering, scaling their work to large tasks can be inefficient. Ours is a more restricted approach which allows for a domain expert to specify the modes using an ER diagram.

We make the following contributions: (1) We propose an approach to make ILP and PLM systems more usable by domain experts by creating a graphical user interface. (2) We demonstrate how effective background knowledge can be encoded using an ER diagram and provide an algorithm for the translation from UI input to a mode specification file. (3) We show empirically the effectiveness of our learning approach in standard PLM tasks.

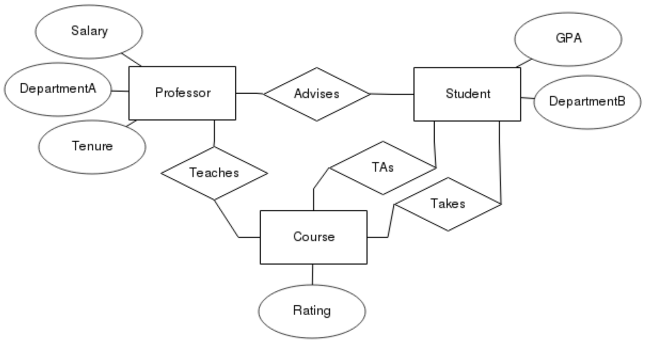


Figure 1: An ER-Diagram illustrating 3 entities, *Professors, Students, and Courses,* their attributes (ovals) and the relationships (diamonds) among them.

We first provide the necessary background knowledge of ER diagrams and ILP systems. Then we outline the procedure for converting the ER inputs to mode definitions. We finally conclude the paper by demonstrating the effectiveness on standard PLM domains and outlining the directions for future research.

2 BACKGROUND

As our approach uses ER diagrams as a way of constructing background knowledge through modes, we discuss both the diagram as well as how modes are typically used in ILP.

2.1 The Entity-Relational Model

The entity-relationship model [3] allows for expressing the structure and semantics of a database at an abstract level as objects and classes of objects (entities and entity classes), attributes of such entities, and relationships that exist between such entity classes. Entities are represented as rectangles, attributes as circles and relationships as diamonds. While relational logic (as used in ILP) is equally expressive, ER models are represented as graphical structures (ER diagrams) making them more intuitive and interpretable. ER modeling is insufficient for expressing operations on the data, but in our problem setting that has no impact. Figure 1 illustrates an ER diagram for an example domain.

ER diagrams are commonly utilized by both database designers and domain experts to conceptualize the structural characteristics of a given domain [5]. A relational schema is an alternate abstract representation of the structure of relational data consisting of structural definitions of relational tables, attributes, foreign key constraints, etc. Conceptually, the knowledge conveyed by a relational schema can be used for abstracting background/modes for an ILP/PLM learning task. But, the limitation lies in the ambiguity it may introduce. For instance, the relation node *"TAs"* (Figure 1) can be expressed in a relational schema either as a foreign key constraint from one entity to another or as a table with 2 columns having the unique identifiers of the connected entities *Course* and *Student.* The choice typically depends on the design of the system that will use the database. ER diagrams avoid such ambiguity via consistent syntax.

Our approach is motivated from the intuitive connections between constrained logical clause search and SQL (Structured Query Language) query augmentation. Logical clauses are equivalent to relational queries since, fundamentally, SQL statements are manifestations of entity sets defined via relational calculus. Several ILP/PLM learning frameworks have successfully utilized this concept [11, 16]. Similarly, modes for clause learning can be interpreted as constraints on relational query construction and query evaluation. "Hints," in relational queries, are special symbolic tools to guide the query evaluation engine to prioritize some database operation over the others to enhance efficiency [2, 4, 10]. Thus, they are akin to soft directives/constraints (modes) on the search space.

2.2 Background Knowledge for ILP

Background knowledge serves two purposes in ILP systems: describing the underlying structure of data and constraining the space of models over which the algorithm explores. Thus, background knowledge (set via modes) is a key component for getting relational learning algorithms to work effectively. A mode describes a way of instantiating predicates in a clause that defines a hypothesis. A mode for predicate $pred$ with n arguments is defined as $pred(type1, type2, ..., typen)$. Each type describes the domain of objects which can appear as that argument, as well as whether it can be instantiated with an input variable (+), an output variable (-), or a constant (#) [20]. Input variables must be previously defined in the model. Output variables are free variables that have not been defined.

ILP learners search through the space of models in different ways. Aleph [20] generates clauses bottom-up by constructing the most specific explanation of examples and then generalizing while TILDE [1] constructs clauses top-down. Our mode construction approach is capable of generating background knowledge for a variety of different ILP systems. To validate our approach, we make use of a state-of-the-art ILP system called Relational Functional Gradient Boosting (RFGB) [14] that learns a set of boosted relational regression trees in a top-down manner. Relational regression trees contain relational logic in the inner nodes and regression values in the leaves. Each iteration of RFGB learns a tree (ψ_k) that pushes the model in the direction of the current error. The error of the current model (Δ_{k-1}) is computed over each training example: $\Delta_{k-1}(x_i) = I(y_i = 1) - P(y_i = 1|Pa(x_i))$ where I is an indicator function for whether x_i is a positive example and P represents the current predicted probability. The final model is a sum over all of trees ($\psi_M = \psi_0 + \psi_1 + ... + \psi_m$). For more details we refer to Natarajan et al. [14].

3 HUMAN GUIDED MODE CONSTRUCTION

Naive approaches for mode construction may allow for exhaustive search, enabling the ILP learner to find the best solution at the cost of a time intensive search process. Other approaches allow for one free variable for each atom considered. This restricts the search space, but ignores the fact that not all areas of the search space are equally important for a given target.

Alternatively, we consider guided construction of modes (GMC) for ILP where the human is assumed to be a domain expert and not an ILP expert. The expert is provided the structure of the domain

Algorithm 1 Guided Mode Construction (GMC)

```
 1: procedure GMC(Expert $E$, max depth $d$)
 2:     target $t$, related attributes or entities $\mathbf{I}$ = INTERFACE($E$)
 3:     Modes $\mathbf{M} = \emptyset$
 4:     for $i \in I$ do
 5:         Paths = FINDPATHS($t, i, d$)
 6:         for $p \in$ Path do
 7:             $\mathbf{M} = \mathbf{M} \cup$ CREATEMODE($p$)
 8:         end for
 9:     end for
10:     return $M$
11: end procedure

12: procedure FINDPATHS(target $t$, related attribute/entity $u$, find shortest path $isShortest$, max depth $d$)
13:     $Solutions = \emptyset, Searched = \emptyset, ToExplore = \{t\}$
14:     while $|ToExplore| > 0$ && $len(ToExplore.peek()) < d$ do
15:         $\mathbf{n} = \{x_1, r_1, x_2, r_2, ..., r_{k-1}, x_k\} = ToExplore.dequeue()$
16:         for $r \in \mathbf{R}_{x_k}$ do
17:             for Entity $y \neq x_k$ appearing in relation $r$ do
18:                 if $\{\mathbf{n}, r, y\} \in Searched$ then
19:                     continue
20:                 end if
21:                 if $y == u || u \in \mathbf{A}_y$ then
22:                     $Solutions.append(\{\mathbf{n}, r, y\})$
23:                     if $isShortest$ then
24:                         return $Solutions$
25:                     end if
26:                 end if
27:                 $ToExplore.enqueue(\{\mathbf{n}, r, y\})$
28:             end for
29:         end for
30:     end while
31:     return $Solutions$
32: end procedure

33: procedure CREATEMODE(Path $p$)
34:     Modes $M = \emptyset$
35:     for $\{x_i, r_i, x_{i+1}\} \in p$ do
36:         for Term $t_j \in r_i$ do
37:             if $t_j == e_i$ then
38:                 $t_j = +e_j$
39:             else if $t_j \in \mathbf{A}$ then
40:                 $t_j = \#e_j$
41:             else
42:                 $t_j = -e_j$
43:             end if
44:         end for
45:         $M.append(r_i(t_0, t_1, ..., t_n))$
46:     end for
47:     return $M$
48: end procedure
```

in a graphical user interface that allows the expert to interact with the Entity-Relationship diagram. The target entity about which the model will be learned is marked and the expert is responsible for marking all of the attributes which are relevant to the target. Then, we find paths through entities and relations that are able to connect the target feature with all of the related entities[1] and their attributes. As we describe in more detail later, these paths are the basis for constructing the modes.

3.1 An Illustrative Example

Consider a set of data involving professors, students, and courses, with some associated attributes and relationships between each. Figure 2(a) shows such an ERD where *Grade* (marked in red) was identified by an expert as being an important attribute for predicting *Tenure* (marked in blue). GMC first connects the target concept to the related concepts by finding paths from one to another in the ER diagram. Figure 2(b) shows two paths that connect *Tenure* to *Grade*.

Once these paths are established, variables can be set as being open (-), closed (+), or grounded (#) based on the order in which entities (variables) appear. Since *Tenure* is the target concept which everything should be learned with relation to, the conversion process begins with `Tenure(+Professor)`.

Modes are added to allow the ILP learner to search along the path. We show each step in one path and the corresponding modes that would be generated in Table 1. Note that the entities and attributes are highlighted in different colors to show which arguments have the same type. The first time a type ($Student, Course$) is introduced along the path, the mode is set to $-$ allowing a free variable to be introduced during the search. Subsequently, appearances of a type have modes that are set to +, forcing a previous variable to be used during search. As *Grade* is an attribute (as opposed to an entity), it will be grounded using the # mode.

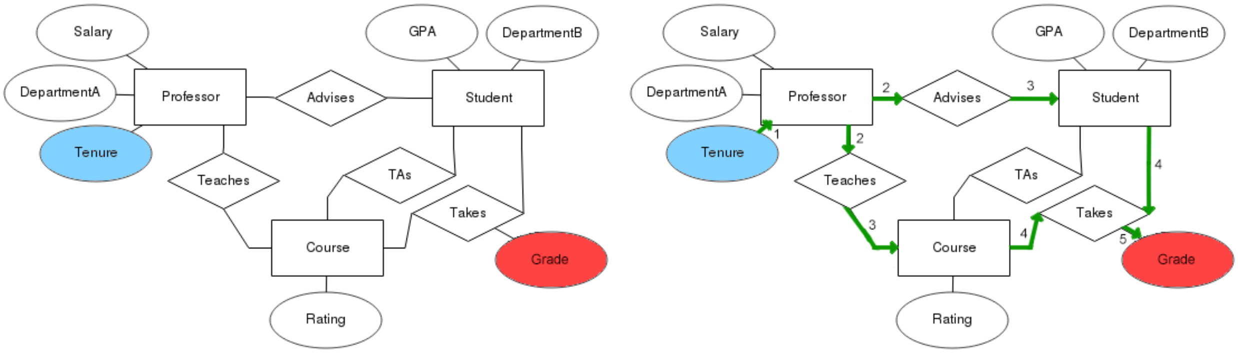
Clause 1 in Table 1 gives an example of a clause that could be generated by an ILP system with the specified modes. The English interpretation of this rule is that tenure depends on the grades of students who are advised by a professor.

3.2 The Algorithm

The goal of GMC (Algorithm 1) is to guide the learner by constructing background knowledge based on input from a human user. This background knowledge consists of a set of modes that defines the search space for an ILP learner, enabling it to find a reasonable hypothesis efficiently. We have created a user interface that allows for a human domain expert to provide relevant attributes or entities for a given target (line **2**). GMC constructs modes that allow these relevant attributes or entities to appear in the model. Thus, the two key steps in GMC are 1) finding paths in the ER diagram (FINDPATHS) and 2) generating modes from those paths (CREATEMODE). We now discuss each of these steps.

3.2.1 FindPaths. Given the target t and a relevant attribute or entity u, we find paths between them in the ER diagram. A path includes the set of entities and relationships which together relate t to u. Each path $p = (t, r_t, x_1, r_1, x_2, r_2, ..., r_{k-1}, x_k)$ consists of attributes or entities ($\{x_i\}$) and relations ($\{r_j\}$). We explore the set of all paths in a breadth first manner starting from t. At each step, we select from among the shortest paths to expand. Assume x_k

[1] Note that ERDs represent entity sets/classes/types and not actual instances or entities to be precise. However, since in the context of our approach we never deal with instances, we use the term "entity" to denote entity classes for brevity



(a) *Target*: 'Tenure'. *Informative/important*: 'Grade', selected by user.

(b) The equidistant shortest paths between Tenure and Grade.

Figure 2: Illustrative example showing knowledge guided walks on the ERD, given in Figure 1, for mode construction.

	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
Path 1	$Tenure$	$Professor$	$Advises$	$Student$	$Takes$	$Grade$
Modes 1	$Tenure(+Prof)$		$Advises(+Prof, -Stud)$		$Takes(+Stud, -Course, \#Grade)$	
Clause 1			$Tenure(p) \wedge Advises(p, s) \wedge Takes(s, c, A+)$			

Table 1: Each step of a path and corresponding modes generated by GMC.

is the current end of the selected path. We denote $\mathbf{R}_{x_k}$ as the set of relations in which entity x_k appears. Path p is then extended for each relation $r \in \mathbf{R}_{x_k}$ by creating a path for each entity that appears in r.

GMC finds a path when it reaches u (if u is an entity) or when it reaches an entity y for which u is an attribute ($u \in \mathbf{A}_y$). There are two potential settings corresponding to the number of paths to be found. If *isShortest* is set to *true*, it will find a shortest path. Otherwise, it will find all paths up to a particular depth d. Our hypothesis is that finding all paths will yield background knowledge that encompass the best model while finding the shortest path will yield the most efficient set of modes that still allow the learner to find acceptable models. Note that the shortest path can be considered the most simple way to relate t and u and such simple knowledge is the basis for many learning algorithms.

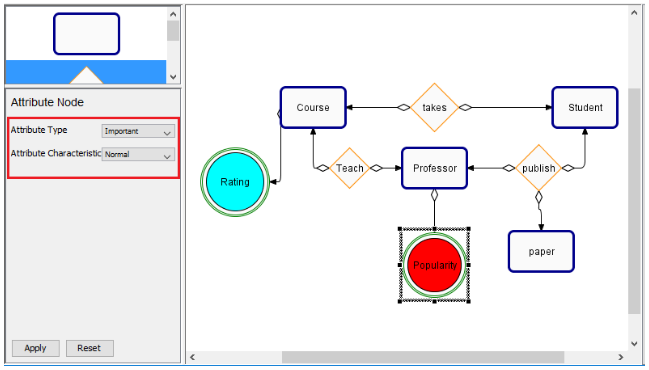
3.2.2 CreateModes. Given a single path p found by FindPaths, we now create a set of modes that will guide the search. As described previously, a mode is specified for a particular predicate. Each argument of the mode specifies the attribute type (defined by the structure of the ER diagram) as well as how new variables/constants can be introduced. For each relationship in the path p, we define a new mode. As mentioned earlier, the number and types of the arguments are defined by the structure of the ER diagram. We also assume that arguments corresponding to attribute values (e.g. the value of blood pressure or grade in a course) are considered as constants (#). Thus, we only need to describe selecting between input/output variables.

For each pair of relations in the path connected through an entity $((r_i, x_{i+1}, r_{i+1}) \in p)$, we generate a mode $m_{r_{i+1}}$ for r_{i+1}. We denote $r_k^{x_j}$ to be the argument of relation r_k that has associated type x_j. We set the argument $a = r_{i+1}^{x_{i+1}}$ as an input variable. All other arguments are set as an output variable ($\forall_{y \in Args(r_{i+1}) \backslash a}\ r_{i+1}^{y}$). Note that there could be multiple arguments with the same type ($|a| \geq 0$). If there are more than one, we generate a mode for each argument in a as an input variable and treat all others as output variables.

The set of modes generated by GMC (M) can be used directly for ILP search. As GMC allows for the domain expert to only provide input on the ER diagram, no expertise in mode construction is required. We now describe our interface in more detail.

3.3 The Interface

The primary objective of our approach is to facilitate a domain expert, having limited understanding of ILP, in creating suitable modes as per the given problem. This necessitates an intuitive and user-friendly interface. We have developed a GUI (Figure 3) that provides a user, having basic understanding of entities and relations, with the tools to build ER diagrams from scratch or load existing ones and annotate them with knowledge about targets and informative attributes/entities. The interface is designed for allowing the user to drag shapes and arrows to construct nodes and edges of a ER diagram as well as to select any node by double clicking on it to set its properties from the drop-down menus in the left pane. The properties include (1) whether the relation/attribute node is the ***target*** (2) whether the attribute/relation is *important/informative/predictive* and (3) if an attribute is multi-valued or binary. Note that if the data is stored in a relational database, an ER diagram can be constructed automatically to some degree of fidelity. But, in most cases, the sanity or the quality of the ER-Diagrams are subject to the database designer's choice.


Attribute Node

Attribute Type: Important

Attribute Characteristic: Normal

Figure 3: The Interface. It provides a drag-and-drop console with drop downs for annotating the ERD. As mentioned earlier: rectangles, diamonds, and ellipses/circles represent entities, relations, and attributes respectively. Here, "Rating" is annotated as the *target* and the "Popularity" attribute is annotated as *important*.

4 EXPERIMENTS

We pose the following questions to evaluate the effectiveness and efficiency of our approach (GMC) in background knowledge construction.

- **Q1**: Does GMC facilitate the learner to optimally explore the hypothesis space (performance)?
- **Q2**: Can GMC enhance efficiency via sufficiently constraining the search space?
- **Q3**: Do simple (`Shortest Path`) modes generate robust models?
- **Q4**: What is the importance of human guidance?

We discuss our results based on two scenarios, (1) searching all paths from our target to our predicates, and (2) exploring the shortest paths. We compare our approach against two baselines: (a) modes encoded by an ILP expert[2], and (b) mode construction based on depth-restricted random paths.

Note that *a* achieves similar performance to walking from the target to every feature in the domain at a lower average training time. *b* is inspired by the success of random walk algorithms that are capable of solving many challenging tasks [9].

The system has two components, (a) a platform-independent GUI component for creation and annotation of ER diagrams and (b) the mode construction from the annotated diagram which is designed to be compatible with any ER diagramming tool given a common intermediate representation.

We have used the state-of-the-art ILP structure/parameter learning framework Relational Functional Gradient Boosting [15] as the test-bed for evaluating the quality of automatically constructed modes.

4.1 Domains

We use four standard ILP/PLM datasets, namely CiteSeer, WebKB, Cora, and IMDB, for an empirical evaluation of our automatic mode construction system. *"facts"* refers to the evidence (all the relations between different objects that are true in the given domain) and *"examples"* refers the total number of positive and negative (true and false) target relations/attributes/features across each cross-validation fold.

CiteSeer [18] dataset was created for information extraction and citation matching. It has 121,891 facts and 116,679 examples split across four cross-validation folds, each corresponding to a different topic. Our goal was to predict which field the title of the paper corresponded to (`infield_ftitle`), and the fourteen other predicates are based on tokens and their relative positions in a document.

WebKB [12] is a consolidated dataset of links among departmental web pages from four universities (Cornell University, University of Texas, University of Washington, and University of Wisconsin) each grouped into one of four cross-validation folds. It has 1912 facts and 747 examples, where the target is to predict `faculty` based on several predicates (`courseProf`, `courseTA`, `project`, and `samePerson`).

Cora [18], like CiteSeer, is about citation matching, with the key difference being the type of relations that are captured. The dataset consists of 6,541 facts and 62,715 examples split into five cross-validation folds. The target is to predict if 2 citations have the same author (`sameAuthor`).

IMDB [12] represents relations between movies and the people who work on them, as well as several attributes of such movies and people. People can either be an actor or a director (mutually exclusive), and the goal is to predict whether an actor worked under a certain director (`workedUnder`). In total there are 664 facts and 5794 examples.

UW-CSE is an anonymized representation of the staff and students of five computer science departments distributed across five cross-validation folds; consisting of 5121 facts and 94,000 examples. The goal is to predict who advises whom (`advisedby`).

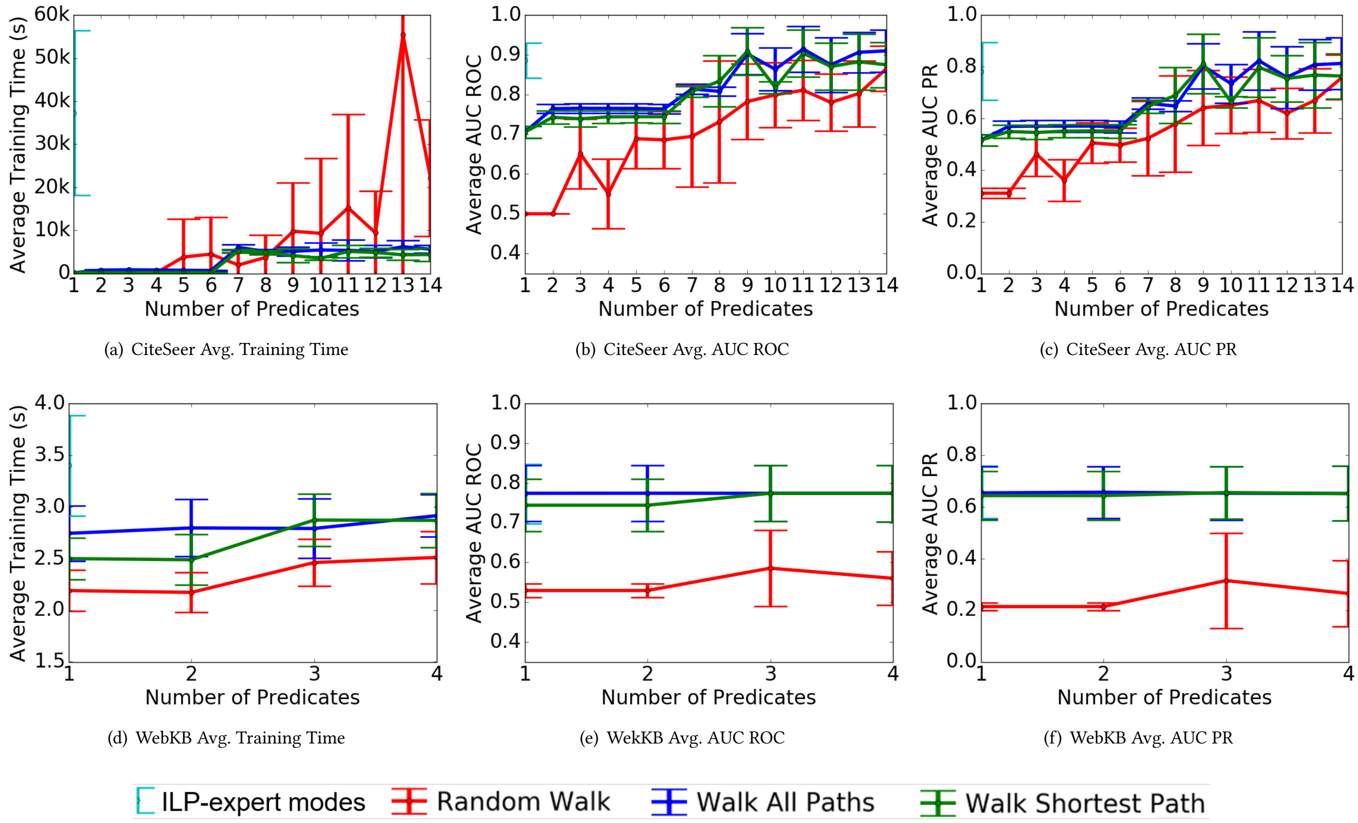
4.2 Experimental Setup

Our GMC algorithm has two settings, `Walk All Paths` for walking all paths on the graph from the user-specified target to each selected feature, and `Walk Shortest Path` for finding only a shortest path from the target to each selected feature.

Experiments were performed on a server with twenty Intel Xeon E5-2690 CPUs clocked at 3.00GHz with no other processes on the server which might interfere with training time. To compare performance for each method, we report the mean and standard deviation of the *training time*, *AUC ROC*, and *AUC PR* across 5 cross-validation folds and 10 independent runs for every dataset and number of features. The settings (namely: negative:positive ratio and #literals at each tree-node) of the underlying PLM learner, 'RFGB', were kept consistent across all the evaluated approaches and 10 trees were learned in each case.

Features (attributes/relations) the expert annotates as important/informative are arranged in the order in which they were selected. In the experimental results (Figures 4, 5 & 6), the **x-axis** represents this ordering, and the respective values for each point represents the performance of a horizontal slice of all predicates up to and including that point. This shows how each additional predicate influences performance/training time.

[2] Discussion on manual mode construction is beyond the scope of this paper.



(a) CiteSeer Avg. Training Time

(b) CiteSeer Avg. AUC ROC

(c) CiteSeer Avg. AUC PR

(d) WebKB Avg. Training Time

(e) WekKB Avg. AUC ROC

(f) WebKB Avg. AUC PR

Figure 4: Results for CiteSeer and WebKB Datasets; Top row: Citeseer, Bottom row: WebKB. Left: Efficiency - Training time (lower is better), Middle & Right: Performance - Average AUC ROC and AUC PR respectively (higher is better).

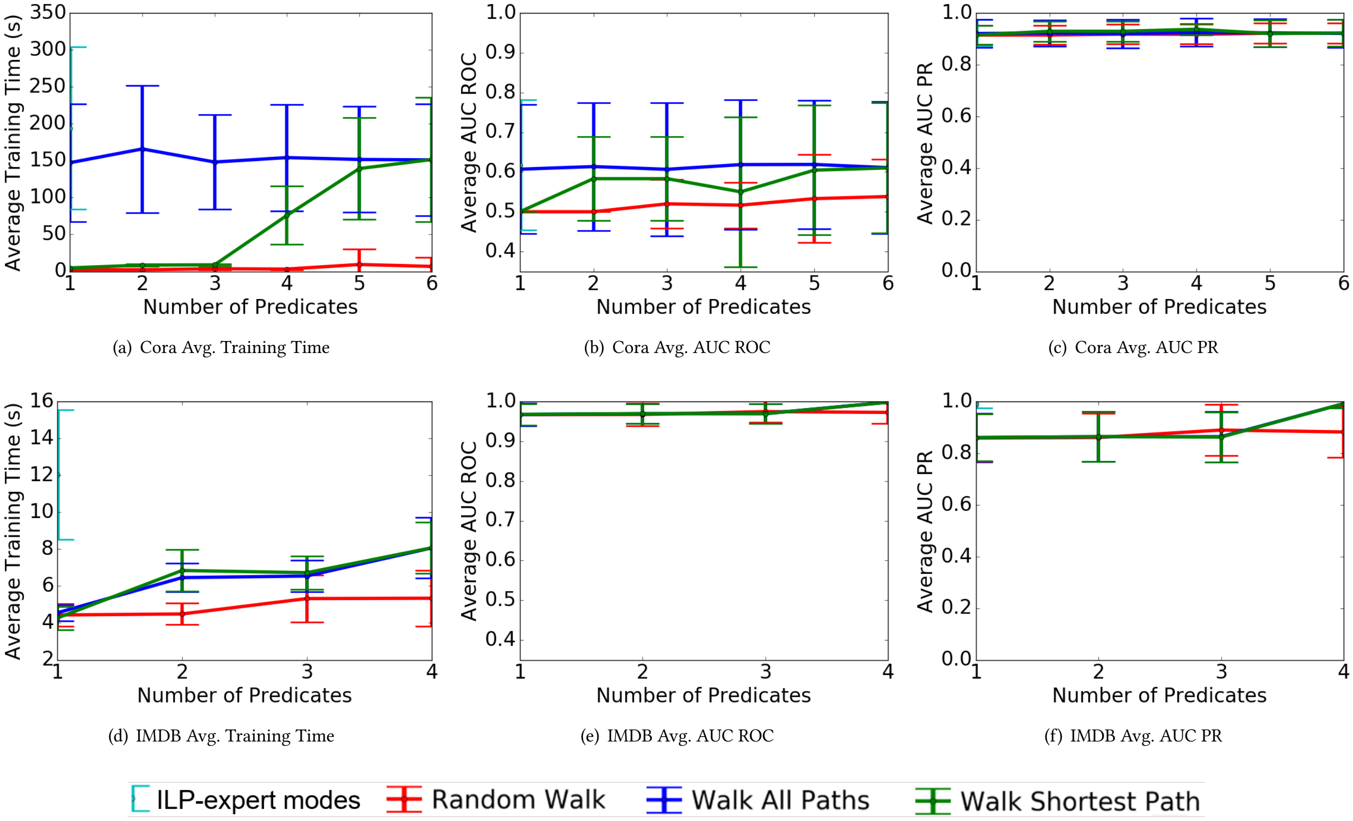
4.3 Experimental Results

From Figures 4, 5 and 6 we observe that both settings of our GMC algorithm outperforms *Random Walk* in 3 of the datasets (CiteSeer/WebKB in AUC ROC and CiteSeer/WebKB/UW-CSE in AUC PR). The difference is more pronounced earlier in the learning curve when fewer paths are being found. As expected, when the number of paths increase, the performance of *Random Walk* often approaches GMC. Both of our GMC approaches are capable of matching the performance of *ILP-expert modes*, often with very few informative predicates marked (except in the case of CiteSeer which requires additional marked predicates). Thus, our GMC methods generate modes that facilitate effective ILP search (**Q1**).

While our GMC approaches generate high performance, they also constrain the search space to allow for efficient models to be learned. The training time of *Random Walk* varies: it is lower than our approaches in three datasets (WebKB/Cora/IMDB) and higher in CiteSeer and UW-CSE. Even though *Random Walk* is more efficient, it is less effective (WebKB/Cora) than our approaches. When compared to the *ILP-expert modes*, our GMC approaches are significantly more efficient in all domains except Cora, where *Walk All Paths* performs similarly to *ILP-expert modes*. Overall, both of our GMC approaches are capable of learning more efficient models than the baseline while also achieving high performance (**Q2**).

While both variants of our GMC algorithm (*Walk All Paths* and *Walk Shortest Path*) compare favorably to the other baselines, we now discuss their differences. Intuitively, *Walk Shortest Path* should have an efficiency advantage over *Walk All Paths*. This is demonstrated in two domains (WebKB/Cora) where *Walk Shortest Path* achieves similar performance to *Walk All Paths* while having significantly lower training time. In all other domains, both variants perform similarly. This suggests that the shortest explanation is often sufficient and allows for a robust and efficient search (**Q3**).

To better comprehend the role of human guidance (**Q4**), let us consider two—not necessarily distinct—aspects. Primarily, human guidance acts as search space constraints for the ILP learner to efficiently search for models. Hence, careful encoding of modes is necessary to achieve comparable, at times better, performance than a super-exponential exhaustive search. *Random Walks* can manage to reduce the search space by working with randomly sampled regions. However, as the results illustrate (Figures 4(e), 4(f), etc.), it may not result in robust models. The other aspect is knowledge about what the most important features/nodes are in automatic mode construction. The empirical results illustrate that, across all datasets and all empirical measurements, there exists a convergence



(a) Cora Avg. Training Time

(b) Cora Avg. AUC ROC

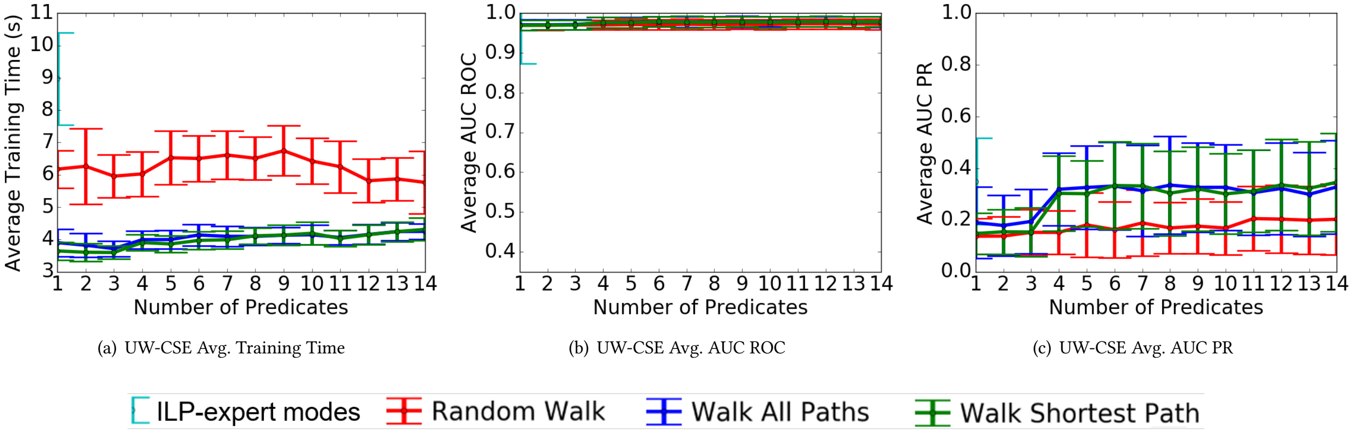
(c) Cora Avg. AUC PR

(d) IMDB Avg. Training Time

(e) IMDB Avg. AUC ROC

(f) IMDB Avg. AUC PR

Figure 5: Results for Cora and IMDB Datasets; Top row: Cora, Bottom row: IMDB. Left: Efficiency - Training time (lower is better), Middle & Right: Performance - Average AUC ROC and AUC PR respectively (higher is better).



(a) UW-CSE Avg. Training Time

(b) UW-CSE Avg. AUC ROC

(c) UW-CSE Avg. AUC PR

Figure 6: UW-CSE results. Left: Efficiency - Training time (lower is better), Middle & Right: Performance - Average AUC ROC and AUC PR respectively (higher is better).

point where including additional guidance (annotations of important features) no longer leads to better performance. *IMDB* requires all four predicates to be taken into account; but in *CiteSeer*, *WebKB*, and *Cora*: performance no longer improves after predicates **9**, **3**, and

1, respectively. In all cases except *Cora*, training time continues to increase slightly while overall performance stabilizes. The domain expert is essential for providing the initial ER model as well as annotating what the most important features/nodes are.

5 CONCLUSION

We considered the problem of capturing domain expert knowledge in the context of learning first-order probabilistic models. We developed a solution based on entity relationship diagrams that allows the domain expert to provide relevant knowledge effectively for making the search process efficient. Our solution is inspired by the observation that most probabilistic logic models can be seen as learning a probabilistic model over a relational graph in the lines of probabilistic relational models [6] and probabilistic entity-relational models [8]. Given this observation, the domain expert identifies relevant nodes in the ER diagram which translates to providing appropriate modes for a clause learning system. Our experiments on standard PLM domains demonstrate the effectiveness of our proposed approach. Extending this system to actively solicit advice as needed [17] is a possible future direction. Allowing for incomplete/noisy and even competing advice is another direction. Finally, extending the interface to allow for knowledge capture in other learning frameworks such as sequential decision-making in relational models, relational deep networks, and other relational models remain an interesting direction for future research.

ACKNOWLEDGEMENTS

Mayukh Das and Sriraam Natarajan gratefully acknowledge the support of the CwC Program Contract W911NF-15-1-0461 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO). Phillip Odom and Sriraam Natarajan acknowledge the support of the Army Research Office (ARO) grant number W911NF-13-1-0432 under the Young Investigator Program.

REFERENCES

[1] H. Blockeel. Top-down induction of first order logical decision trees. *AI Commun.*, 12(1-2), 1999.

[2] N. Bruno, R. Ramamurthy, and S. Chaudhuri. Flexible query hints in a relational database, May 29 2012. US Patent 8,190,595.

[3] P. P.-S. Chen. The Entity-Relationship Model–Toward a Unified View of Data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36, 1976.

[4] A. Diab, S. A. Gatz, S. Kapur, D. Ku, C. Kung, P. Hoang, Q. Lu, L. Pogue, Y. K. Shen, N. Shi, et al. Search system using search subdomain and hints to subdomains in search query statements and sponsored results on a subdomain-by-subdomain basis, Mar. 3 2009. US Patent 7,499,914.

[5] H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database Systems, Second Edition.* Pearson Education, Inc, 2009.

[6] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. *Relational Data Mining, S. Dzeroski and N. Lavrac, Eds.*, 2001.

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

[8] D. Heckerman, C. Meek, and D. Koller. Probabilistic models for relational data. Technical Report MSR-TR-2004-30, March 2004.

[9] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67, 2010.

[10] G. M. Lohman, E. J. Shekita, D. E. Simmen, and M. S. Urata. Relational database query optimization to perform query evaluation plan, pruning based on the partition properties, July 18 2000. US Patent 6,092,062.

[11] M. Malec, T. Khot, J. Nagy, E. Blasch, and S. Natarajan. Inductive logic programming meets relational databases: An application to statistical relational learning. In *ILP*, 2016.

[12] L. Mihalkova and R. Mooney. Bottom-up learning of Markov logic network structure. In *ICML*, pages 625–632, 2007.

[13] S. Natarajan, K. Kersting, T. Khot, and J. Shavlik. *Boosted Statistical Relational Learners: From Benchmarks to Data-Driven Medicine.* Springer, 2015.

[14] S. Natarajan, T. Khot, K. Kersting, B. Gutmann, and J. Shavlik. Gradient-based boosting for statistical relational learning: The Relational Dependency Network case. *MLJ*, 2012.

[15] S. Natarajan, T. Khot, K. Kersting, B. Guttmann, and J. Shavlik. Boosting Relational Dependency networks. In *ILP*, 2010.

[16] F. Niu, C. Ré, A. Doan, and J. Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *CoRR*, abs/1104.3216, 2011.

[17] P. Odom and S. Natarajan. Actively interacting with experts: A probabilistic logic approach. In *ECML*, 2016.

[18] H. Poon and P. Domingos. Joint inference in information extraction. In *AAAI*, pages 913–918, 2007.

[19] L. D. Raedt, K. Kersting, S. Natarajan, and D. Poole. *Statistical relational artificial intelligence: Logic, probability, and computation.* Morgan & Claypool Publishers, 2016.

[20] A. Srinivasan. *The Aleph Manual*, 2004.

[21] T. Walker, G. Kunapuli, N. Larsen, D. Page, and J. Shavlik. Integrating knowledge capture and supervised learning through a human-computer interface. In *KCAP*, 2011.

[22] T. Walker, C. O'Reilly, G. Kunapuli, S. Natarajan, R. Maclin, D. Page, and J. Shavlik. Automating the ILP Setup Task: Converting User Advice about Specific Examples into General Background Knowledge. In *International Conference on Inductive Logic Programming*, pages 253–268. Springer, 2010.

[23] Q. Zeng, J. M. Patel, and D. Page. Quickfoil: scalable inductive logic programming. *Proceedings of the VLDB Endowment*, 2014.