2021

Explanation Techniques using Markov Logic Networks

Khan Mohammad Al Farabi

Explanation Techniques using Markov Logic Networks

by

Khan Mohammad Al Farabi

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Science

Major: Computer Science

The University of Memphis

October 2021

Acknowledgements

First of all, I would like to thank my advisor, Dr. Deepak Venugopal, for his support and intellectual guidance throughout the path towards the successful completion of the Ph.D. It is my privilege to exchange intuitive ideas and receive productive feedback from him during my Ph.D. His advice on research, as well as in my career, has been invaluable. In addition, my special thanks to my Ph.D. Committee members Dr. Vasile Rus, Dr. Amy Cook, and Dr. Xiaofei Zhang who sacrificed their scarce time.

I also want to acknowledge NSF (IIS Grant #2008812) for supporting me during my Ph.D. I would also like to thank Dr. Sanorita Dey and Dr. Somdeb Sarkhel for their valuable suggestions and time in this dissertation research. Finally, I want to recognize my parents' sacrifice. I want to dedicate this dissertation to all of them who sacrificed their valuable time and effort to help me to complete my Ph.D. successfully.

Abstract

Explaining the results of Artificial Intelligence (AI) or Machine Learning (ML) algorithms is crucial given the rapid growth and potential applicability of these methods in critical domains including healthcare, defense, autonomous driving, etc. While AI/ML approaches yield highly accurate results in many challenging tasks such as natural language understanding, visual recognition, game playing, etc., the underlying principles behind such results are not easily understood. Thus, the trust in AI/ML methods for critical application domains is significantly lacking. While there has been progress in explaining classifiers, there are two significant drawbacks. First, current explanation approaches assume independence in the data instances which is problematic when the data is relational in nature, which is the case in several real-world problems. Second, explanations that only rely on individual instances are less interpretable since they do not utilize relational information which may be more intuitive to understand for a human user. In this dissertation, we have developed explanations using Markov Logic Networks (MLNs) which are highly expressive statistical relational models that combine first-order logic with probabilistic graphical models. Since MLNs are symbolic models, it is possible to extract explanations that are human-interpretable. However, doing this is computationally hard for large MLNs since we need to perform probabilistic inference to attribute the influence of symbolic formulas to the predictions. In this dissertation, we have developed a suite of fundamental techniques that help us in i) explaining probabilistic inference in MLNs and also ii) utilize MLNs as a symbolic model for specifying relational dependencies that can be used in other explanation methods. Thus, this dissertation significantly advances the state-of-the-art in explanations for relational models, and helps improve transparency and trust in these models.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine Learning (ML) has contributed to significant advances in several AI domains including natural language understanding, computer vision and game playing. However, with the growing use of ML in virtually every conceivable domain, ML-based methods can no longer remain"black-boxes", i.e., methods that yield highly accurate results on tasks without revealing the reasoning behind their results. Indeed, for ML-methods to be truly useful in critical domains such as health care, it is important to be able to explain underlying logic behind an ML method. Methods such as Deep Neural Networks (DNNs) have achieved unprecedented, state-of-the-art results in several challenging domains such as natural language processing, computer vision, game playing, etc. However, DNNs are known to be extremely hard to interpret which is problematic in several domains. At the same time, Statistical Relational Models [21] are models that are based on symbolic AI (e.g. first-order logic based models) and are therefore much more interpretable in general. In this dissertation, we have developed explainable methods for Markov Logic Networks (MLNs) [15], arguably the most popular Statistical Relational Model.

Markov Logic Networks (MLNs) [15] are popular Statistical Relational Models that combine first-order logic with probabilistic graphical models [39]. The power of MLNs comes from the fact that they can represent relational structure as well as uncertainty in a highly compact manner. Specifically, an MLN represents real-world knowledge in the form of weighted first-order logic formulas. Unlike traditional first-order logic based representations, MLNs allow uncertainty in the represented knowledge, where weights attached to the formulas encode this uncertainty. Larger weights indicate more belief in a formula as compared to smaller weights.

Further, an MLN acts as a template from which we can generate large and varied probability distributions that encode the uncertainty inherent in most real-world applications. Specifically, the formulas in an MLN can be instantiated with objects from a real-world domains, and then converted into potential functions in a Markov network (an undirected probabilistic graphical model). The Markov network represents a joint distribution over various *possible worlds* of the MLN. Note that based on the structure of the formulas in the MLN, several types of well-known models can be represented using this approach including Hidden-Markov Models, Logistic Regression models, etc. Also, the underlying Markov network could be extremely large while the MLN representation is highly compact. In other words, by writing a few formulas in the MLN, we could represent a probabilistic graphical model with thousands of nodes and potential functions by instantiating the MLN with large number of domain objects. At the same time, the MLN representation allows parameter sharing between the vast number of potential functions since these functions are obtained by instantiating the same template (first-order formula) with different types of objects. Thus, MLNs have been applied successfully in varied practical problems such as coreference resolution [62], information extraction [61, 96], question answering [36], event-detection in videos [89], etc.

One of the key advantages of MLNs is their interpetability. Specifically, since MLN models are first-order logic based models, it is quite easy for a human user to understand and interpret what the learned model represents. For example, we can represent some real-world concepts such as smoking causes cancer by relating predicate-symbols the denote smoking and cancer, i.e., $\texttt{Smokes}(x) \Rightarrow \texttt{Cancer}(x)$. This is in contrast to methods such as deep learning where we learn a sub-symbolic model that stores pieces of information distributed in different nodes of the neural network and the resulting encoding of variables is generally a numerical vector that combines this information. However, even though the structure of MLNs are interpretable, there are several challenges in explaining MLNs to humans. Specifically,

1. The parameters of the MLN which are weights attached to the formulas are not probabilities. Quantifying the exact influence of formulas to the results obtained when we perform

probabilistic inference using the MLN is a computationally hard problem.

2. In many cases, the MLN may contain a large number of formulas which makes it hard to learn the parameters of the model efficiently and complex model make it harder to generate explainable inference results.

3. When the MLN distribution has a large number of variables, it becomes harder to generate explanations that are human-interpretable.

Guidotti et al. [25] provide a detailed survey of explanations in ML methods in which they categorize explanations as model explanations and outcome explanations. The former provides explanations for the model (interpretability of the model) while the latter provides explanations for predictions. Of late, there has been a lot of interest in outcome explanations [26]. For instance, in healthcare applications, a doctor would require a system that explains why it is recommending a particular action based on the observations from the patient, rather than just provide results as a "black-box". Some ML methods such as decision trees are both interpretable and explainable, while some are neither (e.g. deep networks). Popular outcome explanation approaches such as LIME [67] try to explain the results of any classifier whose results are typically hard-to-understand. However, the key limitation of such approaches is that they are specific to non-relational data. That is, we assume that each instance in the dataset is predicted independent of other instances in the dataset. Thus, we can generate an explanation for each prediction independently. However, in many cases, real-world data is relational and thus, making the independence assumption is a poor approximation for such data. For example, continuing our health care example, suppose a patient visits a doctor several times, then each visit cannot be considered independently from other visits. Thus, the data has some inherent relational structure and typical machine learning methods ignore this relational structure. In contrast, MLNs explicitly model relational structure in the data. Therefore, instead of classifying instances independently, we perform probabilistic inference jointly over all instances. Thus, the explanations for the inference results take into account the relationships across the instances in the MLN. This leads to much richer explanations as compared to approaches

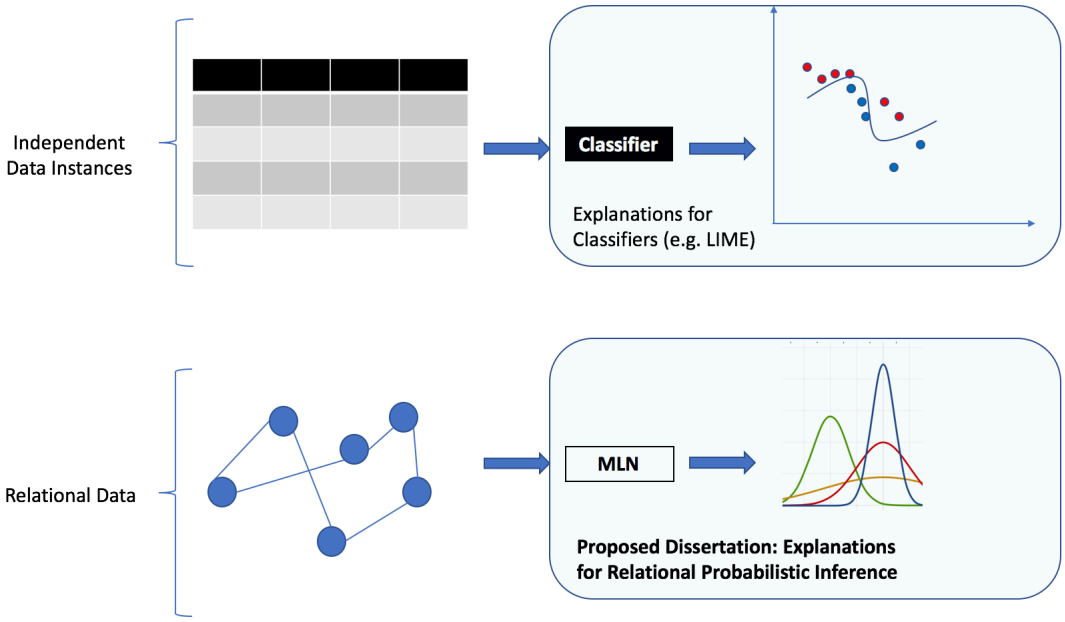


Figure 1.1: Schematic illustration of the our research.

such as LIME.

1.1 Contributions

Our contributions are illustrated schematically in Fig. 1.1. Specifically, similar to methods such as LIME that explain classifiers for standard black-box ML algorithms, our work aims to explain probabilistic inference made from the MLN's distribution. Note that a key distinction between the standard explanations and our approach is that our explanations are based on the relationships across the dataset and therefore the explanations are richer than explanations that ignore this structure.

1.2 Dissertation Work

The specific contributions of this dissertation include the following.

- An approach that learn a simplified approximate model for a high-dimensional MLN using *tied* parameters.

- An explanation algorithm for marginal inference in MLNs that quantifies the effect of formulas on the marginal probabilities based on Gibbs sampling.
- A novel approach for more reliable explanations in large MLNs that explains simpler MLN distributions assuming symmetries in the model and then combines importance weights from several simpler explanations together into a unified explanation.
- A framework for using feedback for relational explanations applied to both model agnostic explainers such as LIME and SHAP [48] as well as model intrinsic explanations using attention mechanisms in deep networks [91].

In chapter 3, we develop an approach to learn a simpler MLN model when the MLN contains high-dimensional, *untied formulas*. Specifically, in MLNs, a single weight is shared across all possible ground formulas. However, in real-world applications, we need to encode high-dimensional features as MLN formulas into an MLN [96]. Such features are encoded using a large number of ground formulas and sharing a single weight across all these features is not expressive enough for real-world applications. Therefore, we use untied formulas to add a different weight for each grounding of a first-order formula. However, now the model becomes extremely large (due to the large number of weights) and infeasible to learn. To simplify the model, we a) reduce the parameter search-space by *tying* together groundings of untied formulas that are likely to have similar weights, and b) perform better initialization for the weights for tied formulas such that discriminative learning (that maximizes the conditional log-likelihood) is likely to converge faster.

In chapter 4, we develop an approach to explain marginal inference by ranking formulas in the MLN. Obtaining such a ranking is computationally hard since we need to estimate the influence of a formula on the marginal probabilities in all possible worlds of the MLN (which is exponential in the number of variables). Therefore, we develop an approach where we compute the importance of the formulas in the transition probabilities of a Gibbs sampler. The importance scores for a formula are then used to select the most important formulas related to a marginal inference query.

We perform a user-study to evaluate the explainability of our results.

When the size of the MLN grows, it becomes increasingly difficult to focus on the correct explanations. Specifically, several formulas may appear to have similar importance weights for a marginal probability since we are approximating this based on samples from the distribution, and for large distributions, we require an infeasible number of samples to obtain reliable explanations. Specifically, the complex MLN increases the mixing time of the sampler that leads to unreliable explanations. Therefore, in chapter 5, we develop an approach that constructs simplified models from large and complex MLN to generate more interpretable explanations. Our approach represents a unified explanation by combining importance weights computed from simplified explanations. We evaluate the quality and interpretability of our explanations using annotated datasets as well as a user study.

In chapter 6, we present a general approach incorporating human-in-the-loop to improve explanations in relational data. Specifically, since explanations are inherently subjective, incorporating a human perspective in explanations can yield more interpretable explanations. We develop and evaluate a framework where we add relational information using MLNs to both model agnostic explanation methods such as LIME and SHAP, and explanations using the attention mechanism in BERT [14], arguably the one of the most well-known deep architectures for language processing. We then propagate feedback for explanations based on symmetries in the relational data and generate more usable explanations. We performed a comprehensive evaluation based on accuracy, information content and usability of the explanation to show the benefits of this approach.

Chapter 2

Background

In this chapter, we present a brief background of Markov Logic Networks (MLNs). Specifically, we describe representation, inference and weight learning in MLNs. For more details on first-order logic, refer to [73, 20], for Markov networks and probabilistic graphical models, refer to [39, 9] and for Markov logic, refer to [15].

2.1 Markov Logic Networks

2.1.1 Propositional Logic

Propositional Logic is a declarative sentence that can be either True or False. There are two types of declarative sentences. One is atomic sentence and the second one is compound sentence. Atomic sentences are only atoms and the compund sentences are the integration of more than one atom through connectives such as $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), $\Longrightarrow$ (implication), and $\Longleftrightarrow$ (equivalence) . These connectives are used to represent the logical relations among the atoms in a compound sentence. For instance,P, Q and R are propositional atoms and f = P $\vee$ Q $\wedge \neg$ R is a propositional formula. A knowledge base (KB) is a set of formulas while a world is a truth assignment to all possible atoms in the KB.

2.1.2 First-Order Logic

A first-order logic can be represented as knowledge base (KB). KB is a set of formulas in first-order logic form [69]. Four types of symbols: constants, logical variables,predicates and logical connec-

tives ($\vee$, $\wedge$, etc.) are used to construct formulas. Variables in first-order-logic are random variables which are substituted by the constants of the domain. The domain of a logical variables refers to all the possible constant values that can be substituted for a variable. We denote the domain of a variable z by Δ_z. Since we assume finite Herbrand semantics [20] the domain-size is finite. Constants are objects in real-world domains and denoted by strings that begin with an uppercase letter (e.g., $Alice$, Bob, etc.). We have considered only a finite subset of first-order logic called *finite Herbrand logic* that has many object constants but no function constants. Logical variables that are denoted by lower case letters (e.g., x, y, z, etc.). A predicate represents relationships between one or more objects. Each predicate has a fixed arity and is represented by a predicate name and a fixed argument-list of variables (e.g., $\texttt{Smokes}(x)$, $\texttt{Friends}(x, y)$, etc.). An atom is an instance of a predicate. A ground atom is an atom where each variable has been substituted by a constant from its domain (e.g. $\texttt{Smokes}(Bob)$, $\texttt{Friends}(Bob, Alice)$, etc.). A formula is a combination of one or more atoms connected by binary connectives (e.g., $\wedge$, $\vee$, $\Rightarrow$ $\Leftrightarrow$), where each variable in the formula is either universally or existentially quantified. In our case, we assume only universally quantified variables and therefore drop the quantifiers for ease of notation. Thus, an example first-order-logic formula is $\texttt{Smokes}(x) \wedge \texttt{Friends}(x, y) \Rightarrow \texttt{Smokes}(y)$. A ground formula is a specific instantiation of a first-order-logicL formula. For example, $\texttt{Smokes}(Bob) \wedge \texttt{Friends}(Bob, Alice)$ $\Rightarrow \texttt{Smokes}(Alice)$. A world is an assignment (either true/false or 0/1) to all ground atoms in the first-order-logic KB. This is typically denoted by ω.

2.1.3 Markov Logic Network

Markov logic networks (MLNs) consist of first-order logic formulas with real-valued weights attached to them. The weight encodes uncertainty in the truth value of the formula. Larger the weight of a formula, more likely is that formula to be true. ∞ weight formulas are treated as hard constraints which should always be true. Similarly formulas with $-\infty$ weights are always false. Thus, MLNs offer a flexible framework to mix hard and soft rules. A predicate is a relation specified in the MLN (e.g., Friends). An MLN formula combines predicates using logical connectives.

A predicate has multiple arguments and the number of arguments define the arity of the relation it defines. In MLNs, we assume Herbrand semantics [20] of first-order logic, where each variable representing an argument in a predicate has a specific type and can substituted by a finite domain of objects (e.g. a set of people in a social-network). We use the symbol Δ_x to denote the domain of variable x. A ground atom is a predicate where all its variables have been substituted by objects from their corresponding domains. We instantiate/ground a first-order formula by replacing each variable in the formula with an object from its corresponding domain. Thus, a ground formula only consists of ground atoms. The ground MLN consists of all possible ground formulas that can be generated given the MLN domains.

An MLN can be seen as a template for generating large Markov networks, the undirected probabilistic graphical model [9, 39], that represents a joint probability distribution over a large number of random variables. We define the Markov network on the ground MLN where each ground formula represents a potential function in the Markov network and each ground atom is a binary random variable (either `True` or `False`). The probability distribution is defined over the assignments to the ground atoms of the MLN and is given by

$$\Pr(\bar{x}) = \frac{1}{Z} \exp\left(\sum_i w_i N_i(\bar{x})\right) \tag{2.1}$$

where w_i is the weight of formula f_i, where $\bar{x}$ is an assignment (of either True or False) to every ground atom in the MLN which is also called a possible world, $N_i(\bar{x})$ is the number of groundings of formula f_i that evaluate to True given world $\bar{x}$, and Z is the normalization constant.

As a simple example of an MLN, suppose we want to encode the fact that smokers and cancer are friends. We would design an MLN with formulas such as a)$\texttt{Smokes}(x) \Rightarrow \texttt{Cancer}(x)$, and b)$\texttt{Smokes}(x) \wedge \texttt{Friends}(x, y) \Rightarrow \texttt{Cancer}(y)$ with weights $w1$, and $w2$ respectively. Here weights,$w1$ and $w2$ encode the uncertainty of the formulas as we have discussed ealier and it is represented as real number. Suppose $\Delta_x = \Delta_y = \{John, Keli\}$. An example ground for-

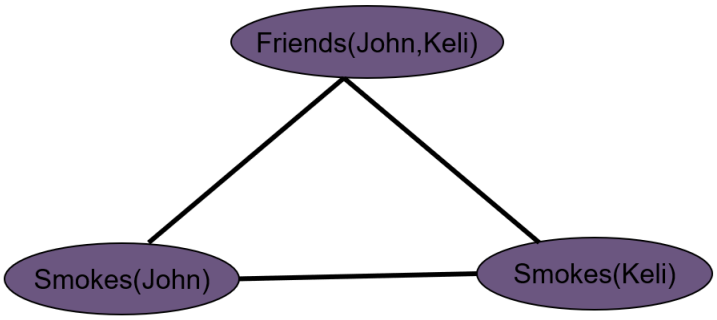


Figure 2.1: Ground Markov Network of formula $\texttt{Smokes}(John) \wedge \texttt{Friends}(John, Keli) \Rightarrow \texttt{Smokes}(Keli)$. Each node in the graph is ground atom and the formula represents a factor in the Markov network.

mula of the MLN is, a) 0.5 $\texttt{Smokes}(John) \Rightarrow \texttt{Cancer}(John)$, and b) 0.6 $\texttt{Smokes}(John) \wedge \texttt{Friends}(John, Keli) \Rightarrow \texttt{Smokes}(Keli)$. The ground atoms in the formula are $\texttt{Smokes}(John)$, $\texttt{Friends}(John, Keli)$, $\texttt{Smokes}(Keli)$, and $\texttt{Cancer}(John)$. Friends predicate defines the relation between John and Keli. Therefore, if Jonh smokes then Keli also smokes. Here, 0.5 and 0.6 are the weights of the formulas that encode uncertainty.

We have constructed the resulting ground Markov network (Fig. 2.1) of the above mentioned ground first-order logic formula where each node in the undirected graph represents a ground atom and the ground formula is denoted by a clique in the Markov network. The neighbors of a node in the ground Markov network is known as the *Markov blanket* for that node. That is a node is conditionally independent of all other nodes given its Markov blanket.

The two key tasks in MLNs are *weight learning*, which is the task of learning the weights attached to the formulas from a relational database, and *inference* (prediction), which is the task of answering queries posed over the learned model given observations (evidence). Weights are learned by maximizing the likelihood of an observed possible world. The marginal inference task which involves finding the marginal probability distribution of a ground atom in the ground Markov network given an evidence database. For example computing the probability that $\texttt{Smokes}(Ana)$ is True given $\texttt{Smokes}(Bob)$ is True and $\texttt{Friends}(Ana, Bob)$ is True,and $\texttt{A}sthma(Bob)$ is false.

This problem is known to be #P-complete and therefore approximate algorithms are used to compute the marginal probability. One of the most widely used inference approach is to approximate the marginals from samples drawn from the MLN's distribution. Among sampling based approximations, Gibbs sampling [19] is arguably one of the most popular approaches for approximate inference.We have discussed details on inference, Gibbs sampling , and learning in the next sections.

2.2 Inference

Inference is one of the major tasks of MLNs. However, to explain the inference in human-understandable form is a challenging issue. In this dissertation, our main focus is to explain the inference in human-interpretable form. In this section, we will become familiar with basic terminologies of inference. We will discuss on propositional inference and lifted inference.The typical inference tasks in MLNs are briefly presented in the following:

- Probability of a query given evidence i.e., $P(Q|E)$ which is known as $Marginal$ inference. For example, given an MLN $(Smoke(x) \implies Cancer(x))$ and evidence telling Ana smokes, i.e., $(Smoke(Ana), \infty)$, we might be interested in knowing posterior probability of Ana has cancer. Computing partition function is equivalent to computing marginal probabilities since marginal probabilities can be expressed as ratio of partition functions.

- Finding a complete assignment to all (non-evidence) ground atoms that maximizes the joint probability distribution over the non-evidence atoms. This is also referred to as finding the most probable state or $Maximum\ a\ Posteriori$ (MAP) inference. Formally,

$$\arg\max_{\omega} \frac{1}{Z} \exp(\sum_i w_i n(f_i, \omega)) \equiv \arg\max_{\omega} \sum_i w_i n(f_i, \omega) \tag{2.2}$$

- Computing the partition function:

$$Z = \sum_{\omega} \exp\left(\sum_{i} w_i n_{f_i}(\omega)\right) \tag{2.3}$$

2.2.1 Propositional Inference

The marginal inference problem and partition function computation problems are known as $\#P$-complete and the MAP inference problem is NP-hard [71]. Inference problems in MLNs are computationally hard. Therefore, exact inference methods are infeasible in practice. One approach to solving inference problems in MLNs is to reduce it to inference in the Ground Markov network. This means that we could potentially apply any known inference technique from Probabilistic Graphics Models (PGMs) to MLNs. For instance, we could apply exact inference methods using dynamic programming such as Variable Elimination [13] or junction trees [41] to solve the inference problems exactly. However, these are feasible only for very simple Markov network structures, namely structures with very low treewidth. In most practical cases, approximate inference methods are used. Popular approximate inference methods include sampling-based approximations or belief propagation (BP). For sampling-based inference, MCMC methods are the most popular methods and among these Gibbs Sampling [19] is perhaps the most widely used approach. A specialized sampler such as MCSAT [63] have been developed for MLNs. Loopy BP methods [99] are an alternative approach based on message-passing in a *factor-graph* constructed from the Markov network. In the case of MAP, stochastic local search methods such as MaxWalkSAT [35] can be used to obtain an approximate MAP assignment. However, all the approaches assume access to the Ground Markov Network which is infeasible in practice.

2.2.2 Lifted Inference

In previous section we have seen that, propositional inference methods don't scale up to Ground Markov Networks represented by MLNs. Therefore, we need more specialized methods that consider relational structure in the MLNs. Specifically, note that the weight in an MLN is shared across

all ground formulas (or potentials in the Markov network). Thus, even though the Ground Markov Network can be extremely large, it is parameterized by much fewer weights. Therefore, the MLN distribution encodes plenty of symmetries. Lifted inference refers broadly to algorithms that exploit these symmetries in the MLNs. For example, consider a very simple MLN, $\texttt{Smoke}(x); w$. Note that, in this case the marginal probability of any ground atom is same. Thus, computing this probability for a ground atom is enough to determine all the marginal probabilities of the objects within the domain of x. Anand et al. [2] identified a sophisticated symmetry in Markov networks and developed MCMC samplers based on these symmetries. The key problem with lifted inference is that symmetries can be efficiently identified only when the MLNs structure is simple. For example, Niepert and Broeck [57] showed that only two variable MLN formulas are liftable. Further, even more importantly evidence breaks symmetries in the MLN [3]. Therefore, performing lifted inference in conditional distributions is hard. Thus, several approaches focus on identifying approximate symmetries in the MLN. To do this, approaches such as smoothing evidence based on binary matrix factorization [3] and clustering-based methods [93, 94] have been developed. Lifted inference which was initially proposed by Poole [60] discovers groups of exchangeable variables in the distribution and performs inference over groups rather than individual variable. Exact inference methods include FOVE [12], WFOMC [90], PTP [23], etc. These methods try to identify symmetries without constructing the Markov network and perform efficient inference using these symmetries. Similarly, approximate inference methods include lifted Gibbs Sampling [92], lifted belief propagation [85], lifted MAP [76], lifted importance sampling [24], etc. In these methods, we identify exchangeable variables and develop approximate inference methods that take advantage of exchangeability. For example, we can reduce the effective sampling-space in sampling-based methods, search-space in MAP methods, etc.

However, MLNs with untied formulas (untied formulas are discussed in the next chapter) make lifted inference infeasible. Further, the accuracy of methods that use approximate symmetries are not very high indicating that, we need richer models to solve more complex problems.

2.3 Gibbs Sampling

Gibbs sampling [19] is one of the most widely used MCMC algorithms to date. Gibbs sampling is used to perform approximate marginal inference in MLNs. In Gibbs sampling, we sample one atom in the distribution at a time given assignments to all other atoms in the MLN. Specifically, given a set of n non-evidence atoms $X_1 \ldots X_n$, the Gibbs sampling algorithm begins with a random assignment $\overline{\mathbf{x}}^{(0)}$ to all non-evidence atoms. Then, for $t = 1, \ldots, T$, it performs the following step (each step is called a Gibbs iteration). We sample a randomly selected atom say X_i from the conditional distribution of X_i given assignments to all other atoms. This is given by

$$P(X_i|\overline{\mathbf{x}}^{(t)}_{-X_i}) = \frac{1}{Z_{X_i}} \exp\left(\sum_f w_f N_f(\overline{\mathbf{x}}^{(t)}_{-X_i} \cup X_i)\right)$$

where $\overline{\mathbf{x}}^{(t)}_{-X_i}$ represents an assignment to all atoms except X_i, Z_{X_i} is the normalization constant. The sample at iteration $t+1$ is $\overline{\mathbf{x}}^{(t)}_{-X_i}$ combined with the sampled assignment to atom X_i.

Gibbs sampling is typically used to estimate the marginal probabilities in the MLN. Typically, the sampler is allowed to run for some time (called the *burnin* time) to allow it to *mix* which ensures that it forgets its initialization and starts to collect samples from the target distribution. After T samples from a mixed Gibbs sampler are generated, the marginal probability of an atom X_i (shorthand for $X_i =$ True) can be estimated using the following equation.

$$\widehat{P}_T(X_i) = \frac{1}{T} \sum_{t=1}^{T} P(X_i|\overline{\mathbf{x}}^{(t)}_{-X_i}) \tag{2.4}$$

We can show that as $T \to \infty$, $\widehat{P}_T(X_i) \to P_T(X_i)$, i.e., as we collect more samples the marginal probabilities converge towards the true marginal probabilities.

2.4 Learning

Learning is an important aspect of MLNs. There are two types of learning for MLNs, a) weight learning and b) structure learning. In weight learning, we are given a MLNs structure with formulas

and evidence and we learn the weights for formulas. In structure learning, we learn the structure of first-order-logic formulas. In this dissertation, we develop a novel approach to make efficient weight learning in MLNs. Therefore, we have briefly represented the basics of weight learning in MLNs.

2.4.1 Weight Learning

Weight learning is a major task of MLNs. We typically use Max-likelihood estimation (MLE) to learn the weights of MLN formulas given MLNs structure and evidence. In general, weight learning we maximize the log-likelihood of the data given a training world, ω. Note that, unlike typical machine learning methods, we have a single instance here from which we need to generalize. Specifically, the log-likelihood is given by,

$$\begin{aligned} \ell(\boldsymbol{\theta} : \omega) = \log P_{\boldsymbol{\theta}}(\omega) &= \sum_i \theta_i N_i(\omega) - \log Z_{\boldsymbol{\theta}} \\ &= \sum_i \theta_i N_i(\omega) - \log \left(\sum_{\omega'} \exp \left(\sum_i \theta_i N_i(\omega') \right) \right) \end{aligned} \tag{2.5}$$

where θ_i is the weight for the ith formula and $N_i(\omega)$ denotes the number of ground formulas in the ith formula that are satisfied by world, ω.

Weights that optimize the log-likelihood (a convex function) can be learned using a standard gradient ascent procedure. However, computing the gradient requires inference in each step which makes learning the optimal weights intractable. Specifically, the gradient is given by (for details, please refer to [15]),

$$\frac{\partial}{\partial w_i} P_w(\omega) = N_i(\omega) - \sum_{\omega'} P_w(\omega') N_i(\omega') \tag{2.6}$$

The ith component of the gradient is the difference between the number of true groundings

of the ith formula in the training database (ω) and its expectation according to the current weights (w). We initialize the weights randomly. In each iteration, we update each weight according to its gradient. However, it is infeasible to compute the expected number of satisfied grounding of a formula. That is, we need to perform marginal inference (i.e., sum over all possible worlds) which is infeasible. Therefore, a standard approach is to approximate the gradient. In voted perceptron [83], we use the MAP assignment to compute the gradient and in contrastive divergence [28, 46], we approximate the gradient using samples. In discriminative learning, we condition over atoms that are guaranteed to be evidence atoms. Thus, we learn optimal weights for the conditional log-likelihood function. Recently, Venugopal et al. [95] proposed a new approach where they approximated the gradient using approximate counting oracles. Specifically, the idea of the gradient direction is enough for weight learning. Therefore, during weight updates they proposed to compute a fast approximation of the samples (or MAP assignment) to scale up learning. Pseudo-likelihood is another approximation which does not require inference and we decompose the likelihood as a product of conditional distributions. However, this typically yields poorer solutions as compared to the other approaches.

Thus, weight learning is computationally hard since each step of gradient ascent requires inference. Moreover, the weight learning in MLNs becomes infeasible when MLNs contained untied formulas. We need a separate weight for each ground formula of an untied formula, and this makes weights learning infeasible. Therefore, in this dissertation we present a novel approach to address this problem.

2.5 LIME Explanations

Locally Interpretable Model Agnostic Explanations (LIME) [67] extracts explanations for a specific prediction made by any *black-box* supervised learning algorithm. Thus, it is a model agnostic approach that can be applied in general to any black-box classifier. That is, even though the classifier as a whole is complex and cannot easily be explained, LIME can explain predictions made by the classifier. The high-level idea in LIME is as follows. Assume that we are given a non-linear

binary classifier $\mathcal{C}$ (this can be extended to multiclass classification as well) and a test instance X to be classified. We want to know which features of X best explain its classification by $\mathcal{C}$. To do this, we perturb X to generate a set of instances $\bar{\mathbf{X}}$. $\bar{X} \in \bar{\mathbf{X}}$ is similar to X but may lie on either side of the decision boundary of $\mathcal{C}$. Thus, $\bar{X}$ represents the local neighborhood of X and each $\bar{X} \in \bar{\mathbf{X}}$ is weighted based on its distance from X. We now learn a simple linear classifier f_X that classifies $X \cup \bar{X}$ such that the classification best matches the one made by the original classifier. f_X therefore represents an approximation of the original classifier $\mathcal{C}$ around the local neighborhood of X. The prediction of $f_X(X) = W^\top X$, where W is a vector of coefficients of the linear classifier. The i-th coefficient is associated with the i-th feature of the classifier. To explain the prediction of X, we rank the coefficients in W where larger values of the coefficients indicate that they are more important to the classification of X as compared to smaller values. To obtain a global overview of all predictions made by the model, LIME also implements a *submodular* pick that picks a diverse sample of instances and explains all these instances such that the user is able to get a unified explanation for the classifier.

2.6 SHAP Explanations

The SHAP explainer [48] is based on Shapely values that are used to assign optimal credit allocation for features in making a prediction. Like LIME, SHAP is also model-agnostic and generates explanations for any black-box model. The main idea in SHAP is to form coalitions, i.e., try to include a feature with subsets of other features and measure the influence of a feature based on all these coalitions. To compute the exact Shapely value is infeasible since the number of subsets of features can be very large. Instead, a popular approach is to use KernelSHAP [48] that approximates the Shapley value. The main idea is to sample coalitons and weight then based on the size of the coalitions. That is, coalitions for a feature with a small number of other features or a large number of other features receives larger weight. This is to ensure that a feature's individual influence as well as its influence in the full classifier are both considered in the explanation. The feature's Shapley estimates in KernelShap can then be derived using a linear regression procedure.

Chapter 3

Efficient Weight Learning in High-Dimensional Untied MLNs

The weight learning is the major task of Markov Logic Networks (MLNs) which is introduced in previous chapter. However, the MLNs with untied formulas (which is discussed later in this chapter) contain large number of ground formulas and each formula has unique weight. Thus, it is infeasible to learn MLNs efficiently and lifted inference becomes very hard due to a) lack of symmetries, b) large number of parameters computation.

Our main contribution in this chapter is an approach to perform efficient and accurate learning in MLNs containing high-dimensional, untied formulas. Specifically, as has been seen in prior applications [96], encoding high-dimensional features as MLN formulas makes learning infeasible, since relational learning tries to learn jointly from the entire training data. Our approach efficiently handles this problem by tying the untied formulas and initializing the formula weights generated by non-relational learner. Naturally, to begin with, we do not know which groundings of untied formulas will turn out to have the same weights. Therefore, we utilize a non-relational learner to determine the effect of each grounding on the classification problem that we are trying to solve. Specifically, we encode an untied formula as a classification problem, extracting i.i.d instances from the relational training database. From this, we derive initialization weights for the groundings of the untied formula, and tie together those groundings that have similar weights. Our underlying assumption is that groundings that exhibit similarity in the non-relational learner will also exhibit similarity in the relational learner. Once we learn the groundings that we want to tie together, we use a relational learner to re-learn the MLN using existing weight learning methods.

We perform experiments on three different real-world problems, namely collective classi-

fication of web-page topics, entity resolution, and information extraction using datasets available in Alchemy [38]. For our experiments, we integrate SVMs with relational learning, and show that our approach yields much more scalable and accurate results, as compared to using state-of-the-art relational learning systems such as Tuffy [58].

3.1 Related Work

In many practical applications of MLNs, researchers have used various strategies to reduce the complexity of learning and inference. Especially when the MLN contains formulas that are hard to learn, alternate techinques have been used to set the weights of such formulas. For example, Venugopal et al. [96] learn high-dimensional linguistic features using SVMs, Khot et al. [36] set some weights manually for hard-to-learn formulas in question answering, etc. Previously, Craven and Slattery [8] proposed an approach to utilize Naive Bayes within propositional rule-based learning through FOIL. Our approach can be seen as an analogous approach but for learning first-order relational models.

On the other hand, there have been a few approaches to perform weight learning in a more efficient manner. Haaren et al. [27] used lifted inference in generative learning where symmetries are better preserved. Recently, Mittal et al. [53] proposed a new approach to learn more fine-grained weights. This approach is similar to ours, but they learn the grouping through hidden variables, and therefore, they need to sum it out using the EM algorithm which is expensive. Particularly, in the presence of high-dimensional, untied formulas, which is our main focus here, their approach is computationally much more expensive, and less scalable as compared to our method. Chou et al [7]proposed a similar idea, where they quantized Bayesian network parameters based on similar values in the CPT. However, in the case of MLNs, learning the original parameters is infeasible, which makes the quantization hard. Sarkhel et al. [77] proposed an efficient discriminative learning approach to learn more efficiently by using approximate counting oracles. But in this case, they assume that the weights of all groundings of a formula are shared. Ahmadi et al. [1] proposed an approach to perform learning in an online manner by using mini-batches of

data. Specifically, they partially ground the MLN using a part of the dataset each time, and update the weights incrementally. However, even with these advancements, applying MLNs to practical applications remains challenging, where we need to encode features that may possibly result in hundreds and thousands of ground formulas, and we would need to perform relational weight learning in an extremely large parameter space. The approach that we have developed in this chapter is an approach to make relational learning applicable to large problems, by integrating it with scalable non-relational learners.

3.2 Weight Learning for Untied MLNs

Given a relational, closed-world training database $\mathcal{D}$, MLN structure $\mathcal{M}$ that encodes high-dimensional untied formulas, and query predicates $\mathbf{Q}$, our task is to learn weights $\mathbf{w}$ for $\mathcal{M}$ that maximizes the conditional log-likelihood (CLL) $P_{\mathcal{M}}(\mathcal{D}|\mathbf{Q})$. Instead of learning a separate weight for each grounding of the untied formulas, we learn weights for a smaller set of formulas by tying together groups of untied formulas. To tie the untied formulas together optimally, we learn which of the untied formulas are likely to have similar weights, using a supervised non-relational learner. We then use weights derived from the non-relational learner as initialization weights when we re-learn the model using a relational learner.

3.2.1 Encoding Untied Formulas

We make the following assumptions about the structure of the untied formulas. All the assumptions are reasonable assumptions, and are typically satisfied when we consider the design of MLNs with untied formulas.

- Each formula is a universally quantified clause.
- $+$ variables are a part of the predicate definition. That is, if an argument of a predicate is defined with the $+$ symbol, then all atoms that occur in the MLN corresponding to that predicate are assumed to have $+$ variables in that argument position.

- Each untied formula contains at least one query atom, and each atom in an untied formula contains at least one $+$ variable. The $+$ variables in query predicates define classes that the MLN is designed to predict. Note that for a binary classification problem $+$ variables are strictly not needed, however for ease of explanation, we add a $+$ variable with domains-size of 2. For the query predicates, there is a mutual exclusion constraint imposed on the arguments defined with the $+$ symbol. Specifically, for every grounding to the remaining variables, one and only one ground atom is true among all the ground atoms obtained by grounding the $+$ variables.

- If an untied formula contains multiple query atoms, one of query atoms is designated as the target atom that we are trying to predict. Each query atom must be the target of exactly one untied formula.

Note that with the last two assumptions mean that, we assume untied formulas ti typically model "features" for a classification task. Note that, if none of the variables in the untied formula are query variables, then, learning separate weights will not make sense, since we assume that the training database contains an assignment to all non-query variables (closed world assumption).

Let f be an untied formula. Let $\mathcal{X}_-$ denote the lifted variables in f, i.e., variables that are not associated will have a $+$ symbol. We divide f into two parts, the target atom of f whose class value we are trying to predict, and the rest of the formula in f denoted by $f^{\bar{Q}}$. Let $\mathcal{X}_+$ be the $+$ variables in $f^{\bar{Q}}$.

Case 1. $f^{\bar{Q}}$ has no query atoms. In this case, all atoms in $f^{\bar{Q}}$ are observable, therefore we can use them directly to learn to classify the target atom. To do this, we consider each grounding to $\mathcal{X}_-$ as an i.i.d instance for our non-relational learner. The features are defined by projecting $\mathcal{D}$ on $f^{\bar{Q}}$ independently for every instance. Specifically, let $\bar{x}_-^j$ be the j-th grounding to $\mathcal{X}_-$, $\bar{x}_+^k$ be the k-th grounding to $\mathcal{X}_+$, and $\bar{f}_{jk}^{\bar{Q}}$ be the formula obtained by grounding $f^{\bar{Q}}$ with $\bar{x}_-^j$ and $\bar{x}_+^k$, we derive the feature vector for the j-th instance, $\mathbf{X}_j$ as,

$$X_{jk} = \begin{cases} 1 & \text{if } \bar{f}_{jk}^{\bar{Q}} \text{ is true in } \mathcal{D} \\ 0 & \text{Otherwise} \end{cases} \tag{3.1}$$

Case 2. $f^{\bar{Q}}$ has one or more query atoms. In this case, since query atoms are considered unknown, we cannot use the query atoms in $\mathcal{D}$ directly to learn to classify the target of f. Therefore, we define a pipeline processing order for query atoms. That is, we will have predictions for all query atoms in $f^{\bar{Q}}$ before we can classify the target atom of f. Thus, we compute an ordering over the untied formulas $f_1 \ldots f_n$ such that, the predictions for all query atoms in f_k (other than the target of f_k) are available from $f_1 \ldots f_{k-1}$. In the event that such an ordering is impossible for a formula f_i, we drop the query atoms that cannot be predicted from formulas $f_1 \ldots f_{i-1}$, when computing the features for f_i. The feature matrix is computed as in case 1, except that, we do not use the query atoms in $\mathcal{D}$ but instead use the predicted values for the query atoms when computing the feature vector from Eq. (3.1).

Given the feature vector for all instances of f note that we can use any classifier to learn to classify the target of f. The only requirement from the classifier is that it needs to assign a weight for each dimension of the feature vector which can be used to interpret the contribution of a dimension w.r.t the classification of the target. In our experiments, we used multiclass SVMs as our non-relational classifier. However, our approach can plug-in several other classifiers, and is thus a general method to integrate relational and non-relational classifiers. We utilize the scalability of non-relational classifiers for handling high-dimensional formulas, and relational learning for learning dependencies across instances, thus yielding the best of both worlds.

In the case of SVMs, we use the (normalized) coefficients of the learned hyper-planes of the SVM as the initialization weights for our formulas. Specifically, if the target of f has m classes, for i-th feature dimension, we have a vector of weights Θ_i, where the j-th component of the vector, θ_{ij}, is the j-th coefficient of the hyper-plane that distinguishes class i from the rest of the classes of the target. Thus, the i-th grounding to $\mathcal{X}_+$, and the grounding corresponding to the k-th class of the target atom will get the initialization weight θ_{ij}.

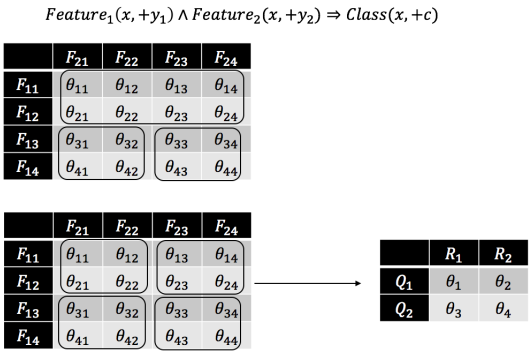


Figure 3.1: Clustering a + variable formula. The first clustering is inconsistent since the atoms of a specific predicate will be clustered with distinct atoms of that predicate for two different clusters. The second clustering is consistent. The θ values specify the initialization weights for the formulas. in the clustered formula, the weight is the average over all weights in that cluster. e.g. $\theta_1 = \frac{\theta_{11}+\theta_{12}+\theta_{21}+\theta_{22}}{4}$.

3.2.2 Clustering

Once we obtain the initialization weights for each grounding of the untied formulas, we reduce the number of formulas by clustering those with similar weights together. Specifically, for every cluster, we replace all the formulas in that cluster with a single formula, with initialization weight equal to the average weight of all formulas in that cluster. However, the clustering must be chosen carefully such that the grouping of formulas is consistent.

For example, consider the formula $\texttt{Feature}_1(x, +y_1) \wedge \texttt{Feature}_2(x, +y_2) \Rightarrow \texttt{Class}(x, +c)$. Let us assume that the domain of y_1 denoted as Δy_1 is equal to $\{F_{11}, F_{12}, F_{13}, F_{14}\}$ and $\Delta y_2 = \{F_{21}, F_{22}, F_{23}, F_{24}\}$. Thus, we have $16 \times |\Delta C|$ formulas. Two different clusterings for this formula are shown in Fig. 3.1. The first clustering is inconsistent, since atoms in two different clustered formulas will overlap. For example, $\texttt{Feature}_1(x, F_{21})$ will be clustered along with $\texttt{Feature}_1(x, F_{22})$, $\texttt{Feature}_1(x, F_{23})$ and $\texttt{Feature}_1(x, F_{24})$ in one cluster, and with only $\texttt{Feature}_1(x, F_{22})$ in a separate cluster. In contrast, the second clustering is consistent. Specifically, along every dimension, the clustered atoms can mapped to a unique cluster.

To achieve a consistent clustering, we model the problem of learning a consistent clustering as a joint clustering problem over all the + variables in the untied formula. Note that we do not consider clustering the + variables corresponding to query predicates. This is because, the +

variables in query predicates correspond to classes, and we want our final model to be able to discriminate between all classes, which is not possible if we cluster the classes.

The joint clustering problem is modeled as follows. Given formula f, let $\mathcal{X}_+$ be the $+$ variables to be jointly clustered. Let $\bar{x}_+$ be a specific assignment to $\mathcal{X}_+$, and let $\theta_{\bar{x}_+}$ be the average initialization weight of groundings of f consistent with $\bar{x}_+$. We arrange $\theta_{\bar{x}_+}$ as a tensor, where each of the $+$ variables represent a single order or dimension of the tensor. Our task is to now find a quantizer that reduces the order of this tensor across all dimensions. Specifically, the quantizer partitions or clusters each dimension of the tensor, to obtain a lower-order tensor. Thus, given the desired number of clusters for each domain corresponding to the $+$ variables, $k_1, \ldots k_m$, the quantizer will jointly partition the domains in $\mathcal{X}_+$.

Formally, let $\mathbf{A}$ be a m-order tensor $\mathbb{R}^{n_1 \times n_2 \times \ldots \times n_m}$ that represents the weights to each possible grounding of $\mathcal{X}_+$. We want to find a quantizer $\mathcal{Q}$ that maps $\mathbf{A}$ to a reduced tensor $\mathbf{B}$, $\mathbb{R}^{k_1 \times k_2 \times \ldots \times k_m}$, that minimizes the Euclidean distance between the cluster-centers in $\mathbf{B}$, and the weights of their corresponding cluster elements. Specifically,

$$\min_{\mathcal{Q}} \sum_{a \in \mathbf{A}} ||a - M_{\mathcal{Q}}(a)||_2^2 \tag{3.2}$$

where $M_{\mathcal{Q}}(a)$ is the cluster center of a under the mapping $\mathcal{Q}$. Note that directly optimizing Eq. (3.2) is a hard problem. Therefore, we use approximate methods that use a co-ordinate descent like procedure and perform dimension-wise clustering as described in [75, 31].

From the reduced tensor for an untied formula f, we reduce the number of possible groundings of f, by utilizing the clusters along each dimension of the tensor. Specifically, given the original tensor $\mathbb{R}^{n_1 \times n_2 \times \ldots \times n_m}$ and the reduced tensor $\mathbb{R}^{k_1 \times k_2 \times \ldots \times k_m}$ induced by the quantizer $\mathcal{Q}$, note that we have two choices. We can generate k_i variables for each dimension, such that the domain of each variable is equal to the objects in the partition specified by $\mathcal{Q}$, and the weight is given by the cluster center. However, though this reduces the number of weights from $n_1 \times n_2 \times \ldots \times n_m$ to $k_1 \times k_2 \times \ldots \times k_m$, the total number of possible groundings of f on the $+$ variables remains $n_1 \times n_2 \times \ldots \times n_m$. This is problematic especially, since we are considering high-dimensional

formulas, the large number of possible groundings makes relational learning infeasible. Therefore, we reduce the number of weights and number of ground formulas at the same time. Specifically, we replace each partition along a dimension with a constant instead of a variable. It then follows that the total number of weights and the number of possible groundings of the $+$ variables, reduces from $n_1 \times n_2 \times \ldots \times n_m$ to $k_1 \times k_2 \times \ldots \times k_m$.

Once we obtain the modified MLN, we perform discriminative relational learning using existing approaches such as voted perceptron and contrastive divergence on the modified MLN to re-learn the weights of all the formulas in the MLN. We can think of this re-learning process as jointly optimizing the weights learned for i.i.d instances. For example, assume that for a given dataset, our non-relational learner gives us ideal weights for $p\%$ of the instances, and imperfect weights for the remaining $(n-p)\%$ instances. Joint learning will now relate the i.i.d instances through the MLN formulas, and jointly optimize their weights to maximize to CLL. Naturally, as p increases, the non-relational learner will influence the relational learner to correct the $n-p$ remaining weights, and as p reduces, the relational learner will use dependencies to correct the weights given by the non-relational learner. Note that in our final learned model, clusters of objects in the domain of $+$ variables will be replaced by constants. Therefore, at testing time, i.e., when we perform inference on this MLN using a test database as evidence, we replace $+$ domain objects in the evidence atoms with symbols that represent their clusters.

3.2.3 Tying Related Formulas

So far, we considered untied formulas independently. However, when atoms are shared across untied formulas, the clustering over one formula affects the other. Specifically, let f and f' be two untied formulas with shared atoms. Clustering the groundings of f affects the formulas in f'. This is similar to the *shattering* process seen frequently in lifted inference [60, 23]. In shattering, grounding the domain of a variable in one formula needs to be propagated to other formulas with a shared domain. In our case, the tying of formulas needs to be propagated to other formulas with one or more shared atoms. Thus, once we cluster a formula, we propagate the clustering

throughout the MLN before clustering the next formula. Propagating the tied formulas means that we replace partitions of the $+$ variable domains with constants. Note that in doing so we remove $+$ variables from an untied formula and replace it with constant symbols.

Algorithm 1 summarizes our approach. The first step is to train a classifier for each untied formula using a non-relational model, using features corresponding to groundings of the $+$-variables in the untied formula. For this, we create independent instances corresponding to each grounding of the non-$+$ variables to create the training data, and learn the parameters of the non-relational model. We then initialize the weights for an untied formula from the learned parameters, and perform clustering to obtain clusters of ground formulas corresponding to the untied formula that have similar weights in the non-relational model. We replace each of these clusters by a single ground formula with weight initialized to be the mean weight of the cluster. We also change the dataset to reflect by replacing objects with their corresponding cluster symbols. We then propagate the changes (from objects to cluster symbols) to other formulas in the MLN that have shared atoms.

3.2.4 Semi-Formal Analysis

Let $\mathbf{w}$ be the weight vector that we would learn if we used the original MLN with untied weights, i.e., we learn a separate weight for every untied formula. Let $\mathbf{w}'$ be the weight vector that we will learn, if we tie the weights of untied formulas with a quantizer $\mathcal{Q}$. We assume that for all the untied formulas grouped together by $\mathcal{Q}$, the difference between the weights that would be learned before formula tying and after formula tying are bounded by ϵ. Specifically, $|w - \mathcal{Q}(w)| \leq \epsilon$, where $w \in \mathbf{w}$ and $\mathcal{Q}(w) \in \mathbf{w}'$ represents the weight of the group to which w belongs as given by $\mathcal{Q}$. Let $\ell(\mathbf{w}, \mathcal{D})$ represent the likelihood when we learn $\mathbf{w}$ and $\ell(\mathbf{w}', \mathcal{D})$, the likelihood when we learn $\mathbf{w}'$ from $\mathcal{D}$. We can show the following result, similar to the result shown in Chou et al. [7].

Theorem 1. *$\ell(\boldsymbol{w} : \mathcal{D}) - \hat{\ell}(\boldsymbol{w}' : \mathcal{D}) \leq 2\epsilon M$, where M is the number of groundings in the MLN.*

Proof.

$$
\begin{aligned}
\ell(\boldsymbol{w} : \mathcal{D}) &= \sum_i w_i N_i(\mathcal{D}) - \log(\sum_{\mathcal{D}'} \exp(\sum_i w_i N_i(\mathcal{D}'))) \\
&\leq \sum_i (\mathcal{Q}(w_i) + \epsilon)(N_i(\mathcal{D})) \\
&\quad - \log(\sum_{\mathcal{D}'} \exp(\sum_i (\mathcal{Q}(w_i) - \epsilon) \\
&\quad (N_i(\mathcal{D}')))) \\
&= \sum_i \mathcal{Q}(w_i) N_i(\mathcal{D}) + \sum_i \epsilon N_i(\mathcal{D}) \\
&\quad - \log(\sum_{\mathcal{D}'} \exp(\sum_i \mathcal{Q}(w_i) N_i(\mathcal{D}') \\
&\quad \exp(-\sum_i \epsilon N_i(\mathcal{D}')))) \\
&\leq \sum_i \mathcal{Q}(w_i) N_i(\mathcal{D}) + \epsilon M \\
&\quad - \log(\sum_{\mathcal{D}'} \exp(\sum_i \mathcal{Q}(w_i) N_i(\mathcal{D}') \\
&\quad \exp(-\epsilon M))) \\
&= 2\epsilon M + \sum_i \mathcal{Q}(w_i) N_i(\mathcal{D}) \\
&\quad - \log(\sum_{\mathcal{D}'} \exp(\sum_i \mathcal{Q}(w_i) N_i(\mathcal{D}') \\
&= \hat{\ell}(\boldsymbol{w}' : \mathcal{D}) + 2\epsilon M
\end{aligned}
$$

□

Thus, from Theorem 1's result, reducing ϵ will reduce the likelihood difference between the two models. Intuitively, to reduce ϵ, we need to predict which of the united formulas will end up with the same weights. Our approach of using a non-relational classifier performs exactly this

prediction by considering instances independently. Essentially, we are decomposing the relational database $\mathcal{D}$ into a set of i.i.d instances D, and our non-relational model learns similarity of weights using D. Our assumption is that these similarities in weights will continue to hold when we relate the i.i.d instances in D through $\mathcal{D}$, thus reducing ϵ. Similarly, having smaller number of formulas will reduce the difference in likelihood, but at the same time, it can increase the error ϵ.

Algorithm 1: Formula Tying

Input: MLN structure $\mathcal{M}$, Training data $\mathcal{D}$, Query $\mathbf{Q}$, Non-relational learner $\mathcal{R}$
Output: $\mathcal{M}'$, $\mathcal{D}'$

Let $f_1 \ldots f_n$ be the untied formulas in $\mathcal{M}$
for *each* f_i **do**
    `// Construct a classifier for` f_i
    Encode each grounding of non-query $+$-variables in f_i as a feature for $\mathcal{R}$
    for *each grounding of non-*$+$ *variables in* f_i **do**
        $X_i = i$-th grounding of non-$+$ variables in f_i
        $y_i =$ Class of query corresponding to X_i
        $X = X \cup X_i$
        $y = y \cup X_i$
    `// Perform non-relational learning`
    $\Theta =$ Parameters of $\mathcal{R}$ learned from (X, y)
    `// Weight initialization for relational learner`
    Initialize f_i with Θ
    $\mathcal{X}_+ = +$ variables in f_i
    $\mathcal{P}^{(1)} \ldots \mathcal{P}^{(m)} =$ Partitions of domains of $\mathcal{X}_+$
    for $e \in \mathcal{P}^{(1)} \times \ldots \times \mathcal{P}^{(m)}$ **do**
        $\theta =$ Cluster center for e
        $\mathbf{C}$ = Map each partition in e to a constant
        `// Replace clustered formulas with a single formula`
        (f',θ) = Replace $+$ variables in f_i with $\mathbf{C}$
        Add (f',θ) to $\mathcal{M}'$
        `// Propagate changes to training data and the other formulas`
        $\mathcal{D}' =$ Replace objects in e with $\mathbf{C}$ in $\mathcal{D}$
        $\mathcal{M}' =$ Propagate C to $f_{i+1} \ldots f_n$
Copy remaining formulas from $\mathcal{M}$ to $\mathcal{M}'$
return ($\mathcal{M}'$,$\mathcal{D}'$)

Weight Initialization. Due to convexity of the CLL function, irrespective of the starting states, in theory, max-likelihood learning will converge to a globally optimal solution given enough iterations. However, the main problem is that performing exact max-likelihood learning is a hard problem. Specifically, the gradient is given by,

$$\frac{\partial}{\partial w_i} \ell(\boldsymbol{w} : \mathcal{D}) = N_i(\mathcal{D}) - \mathbb{E}_{\boldsymbol{w}}[N_i(\mathcal{D})]$$

Computing $\mathbb{E}_{\boldsymbol{w}}[N_i(\mathcal{D})]$, is typically infeasible since it requires exact inference on the MLN. Therefore, relational weight learners approximate the gradient by computing $\mathbb{E}_{\boldsymbol{w}}[N_i(\mathcal{D})]$ approximately. This approximation means that we cannot guarantee convergence to the globally optimal $\boldsymbol{w}$, and depending on the initialization, the weights may converge to different local minima. Good weight initialization is well-known to be effective in avoiding local minima [97], especially when dealing with a large number of parameters, therefore, our weight initialization will typically be effective in high-dimensional untied formulas.

3.3 Experiments

3.3.1 Setup

We evaluated our approach on three real-world applications modeled using MLNs, namely, collective classification of web-page topics [45] (WebKb), entity resolution [84] (ER), and information extraction [61] (IE). For ER and IE, we used the Cora dataset, that consists of 1295 citations of 132 different papers. For WebKb, the dataset consists of 4165 web-pages from 4 different universities. The MLN structure and the data for each of these applications are publicly available in Alchemy [38]. Note that, we chose these MLNs since they are some of the most popular benchmarks for evaluating MLN inference and learning algorithms.

We conducted our experiments on 8GB quad-core machines. We evaluated the performance of our approach regarding accuracy and running time. As our non-relational learner, we used the multiclass SVM implementation in scikit-sklearn. For the relational learner, we used the contrastive divergence implementation in Tuffy [58]. Tuffy is arguably the leading general-purpose learning and inference system for MLNs. We compared our approach (SVMTied) with three other approaches, SVM classifier (SVM), Tuffy with the original MLN with untied weights as is (OrigTuffy), using tied weights with random initial weights (RandomizedTuffy), and evaluated accuracy using cross-validated F1-score.

SVM	OrigTuffy	RandomizedTuffy	SVMTied
0.42	X	0.31	**0.84**

Table 3.1: F1-Score comparison for WebKB.

3.3.2 Results

Our first application predicts topics in web-pages using the WebKB MLN. The untied formula relates words in the document to topics. There is only one query predicate in this MLN, which encodes the seven classes/topics of web-pages. We tried to use Tuffy to learn the MLN with untied formulas, but Tuffy did not work on the original MLN that contains + variable formulas, and timed out after 24 hours. The results for the remaining systems are shown in Table 3.1. As shown here, the weight initialization plays an essential role in improving performance. When we cluster the untied formulas based on random weights, we obtain performance that is worse than using SVMs, which of course ignores relational dependencies. We used the same number of clusters (80 clusters) in both RandomizedTuffy and our approach. Our approach significantly outperforms the other approaches on this application. Lowd and Domingos [45] obtained an AUC score of around 0.8 for this dataset learning a separate weight for each formula. This illustrates that tying could help improve generalization in some cases, where learning too many weights could overfit the model.

The second application performs entity resolution (ER) using the CORA dataset. Specifically, the idea is to predict if fields belong to the same author, venue, title, and finally if two citations match. Thus, there are multiple query predicates in the MLN for this case. The untied formulas use word-based features to predict the queries. Once again, OrigTuffy failed to run on this problem, and we timed out after 24 hours. For both RandomizedTuffy and SVMTied, we used 40 clusters. Our approach once again significantly outperformed the other methods on all queries for this application as shown in Table 3.2. SVMs outperformed using random weight tying, with random weight initialization. Singla and Domingos [84] obtained an AUC score of around 0.91 for SameAuthor, 0.90 for same venue and 0.99 for SameBib. However, Singla and Domingos use domain-specific methods to make the approach more scalable. For example, they apply McCallum

Query	SVM	OrigTuffy	RandomizedTuffy	SVMTied
SameAuthor	0.71	X	0.33	**0.92**
SameTitle	0.63	X	0.54	**0.85**
SameVenue	0.68	X	0.32	**0.79**
SameBib	0.51	X	0.29	**0.78**

Table 3.2: F1-Score comparison for ER.

Query	SVM	OrigTuffy	RandomizedTuffy	SVMTied
InfieldAuthor	0.65	X	0.36	**0.79**
InfieldTitle	0.45	X	0.4	**0.68**
InfieldVenue	0.52	X	0.42	**0.71**
SameCitation	0.48	X	0.28	**0.72**

Table 3.3: F1-Score comparison for IE.

et al.'s [49] canopy approach with TF-IDF to reduce the number of plausible pairs. It is sometime hard to generalize these methods to other problems, and therefore, in our approach, we did not apply these methods. In future, we will explore systematic ways to incorporate such domain-specific knowledge into MLN weight learning in order to improve accuracy.

Our final application performs joint segmentation and citation matching. We use the Information Extraction (IE) MLN and the citeseer dataset for this application. The untied formulas relate word based features with predicting segments of text, and the segmented text is used in predicting if two citations are the same. The segmented fields can be of type author, tile or venue. As shown in Table 3.3, on all the query variables, our method using 35 clusters significantly outperforms the other approaches. The best published result on this dataset is by Poon and Domingos [61]. Specifically, using the joint segmentation (without incorporating entity resolution feedback), Poon and Domingos obtained an F1-score of nearly 94%. However, the MLN we used is a simpler version (also specified in the Alchemy website), since existential quantification that is needed by the Poon and Domingos MLN is not well-supported in current relational learners. Note that, Poon and Domingos mention in their work that they have modified the Alchemy system to run their MLN, which was not publicly available to us.

Varying Number of Clusters. Increasing the number of clusters reduces the quantization error. We verify whether reducing the quantization error results in improved performance in terms of classification accuracy. Specifically, we learn our models with a varying number of clusters.

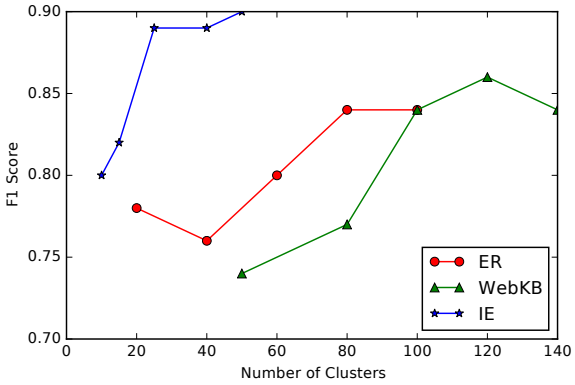


Figure 3.2: Performance of our approach as we vary number of clusters

Application	SVM	OrigTuffy	RandomizedTuffy	SVMTied
WebKB	26	X	200	127
ER	18	X	324	199
IE	20	X	303	145

Table 3.4: Running time (in minutes) comparison.

Fig. 3.2 shows our results where the average F1 score over all queries is plotted against the number of clusters for each of our applications. As can be seen by our results, increasing the number of clusters typically improves performance over all datasets. However, it should also be noted that learning becomes harder as we increase the number of clusters, and the weights learned may not be optimal. Thus, in some cases, we see the performance not improving with an increase in clusters, or in some cases, even degrading by a little. Choosing the optimal number of clusters is something we will consider in future work.

Running Time. We compare the running times of the various learning methods in Table 3.4. As expected, SVMs take very little time to train as compared to relational learners. Note that SVMTied is significantly faster than RandomizedTuffy in terms of training times. This illustrates that proper weight initialization can significantly help speed up convergence of the relational learner.

3.4 Summary

MLNs have potential applications in a number of different domains. However, since MLNs are template models and their representation is quite flexible, it is quite easy to write simple looking

MLN formulas that result in a huge probabilistic graphical model on which learning and inference are infeasible. Particularly, encoding non-relational features as untied MLN formulas typically tends to blow up the size of the MLN, especially if the features have high-dimensionality. In this chapter, we have developed an approach to learn efficiently from such MLNs. Specifically, we tied together MLN formulas that are likely to have similar weights. To predict which of these formulas may have similar weights, we used a non-relational classifier and encoded the learned model as initialization weights for the grounding of untied formulas. We then tied together formulas with similar weights by modeling this as a multi-dimensional clustering problem over variables in the untied formulas and set their initialization weights from the cluster centers. We then re-learned the entire MLN using existing relational weight learning methods. Our experiments on multiple applications showed that our approach significantly improves accuracy and scalability of learning.

However, human users are completely unaware of the internal operation of MLNs model to generate the inference results and as a result, there is a trust issue regarding inference results among human users. In order to address this problem, we have developed an approach to generate human-interpretable explanations of inference in MLNs which is discussed in the next chapter.

Chapter 4

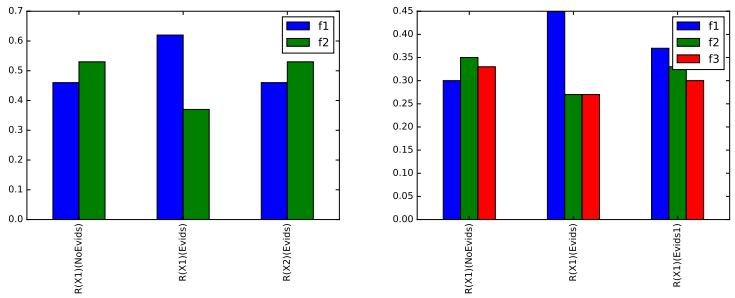
Fine-Grained Explanations using Markov Logic

One of the key advantages of MLNs is their interpetability. Specifically, since MLN models are first-order logic based models, it is quite easy for a human user to understand and interpret what the learned model represents. In contrast, methods such as deep learning can achieve state-of-the-art results in language processing, computer vision, etc., but their lack of interpretability is problematic in many domains. However, interpretability of learned models is not the same as explainability of results generated by the model [25]. Recently proposed approaches such as LIME [67] try to explain the results generated by a model, where the decision boundary of the model is typically complex. Specifically, such approaches approximate the decision boundary by a simpler boundary to explain the classification. However, these approaches are specific to non-relational data, and cannot be applied when the instances are related to each other. Our focus in this chapter is to generate explanations for probabilistic inference in relational data.

It turns out that though MLNs are interpretable, probabilsitic inference in MLNs cannot be easily explained since the distribution is typically large and complex. Our focus in this chapter is to explain relational inference in MLNs in a human-understandable form. Our main idea is to generate explanations for queries in terms of a ranking of formulas based on their importance. Specifically, MLN formulas have weights attached to them that intuitively signifies their importance, i.e., for a formula f with weight w, a world where f is true is e^w more likely than a world in which it is false [15]. Note that the formula weights do not have a well-defined probabilistic interpretation if they are dependent on each other, i.e., if atoms in one formula also occur in other formulas [15]. More importantly, the weights are *tied*, which means that any instantiation of a for-

mula has the same weight. Thus, a naive explanation for a query that can be obtained by ranking formulas purely on their weights is not likely to be useful since it is generic across all possible worlds. That is, the explanation will remain unchanged even when the query or evidence variables change. For example, consider the task of classifying if an email is spam or not. An MLN could encode a formula such as $\texttt{Word}(e, +w) \Rightarrow \texttt{Spam}(e)$. The $+$ symbol in this type formula has been explained in previous chapter. Recalling from previous chapter,this $+$ symbol preceding a variable is a short-hand representation to denote that the MLN stores a different weight for every distinct grounding of the w variable (which represents the domain of words). Suppose the query predicate is `Spam`, we would want different explanations for different groundings of the query predicate based on the specific evidence on the `Word` predicate. Further, suppose the evidence is incomplete, meaning that there are some atoms that are not query atoms and whose truth value is not known. For formulas containing such atoms, it becomes even harder to determine their influence on a query since we need to consider all possible worlds where the unknown atoms are true as well as the cases where the atoms may be false. We have developed a systematic approach for explanations where we learn importance weights for formulas based on samples generated from the MLN. Specifically, we perform inference using Gibbs sampling, and learn the importance of formulas for a specific query based on their influence in computing the Gibbs transition probability. Thus, as the sampler samples possible worlds consistent with the observed evidence, the importance weights capture the influence of formulas on the query variable in these worlds.

We evaluate our approach using two MLN applications we designed for performing inference in real-world review data from Yelp. In the first application, we predict if a review is a spam review and provide explanations for this prediction. In the second application, we predict the sentiment of a review that has missing words. For both cases, we develop MLNs that encode common knowledge and use our approach to extract explanations from the MLNs. We set up a comprehensive user-study consisting of around 60 participants and compare our explanations with explanations given by LIME for the same tasks. We clearly demonstrate through these studies that our explanations are richer and more human-understandable than the explanations given by LIME.



(a) (b)

Figure 4.1: Illustrating the influence of a formula w.r.t a query atom for varying evidences.

4.1 Related Work

Explaining the results of Machine learning models has been recognized as a critical area.LIME developed by Ribeiro et. al. which can provide an explanation of *any* classifier.Most recently, they developed "Anchors" [66], a model-agnostic explainer with *if-then* rules. Ross et al. [70] developed a regularizer to obtain simpler explanations of a classifier's decision boundary. Koh and Liang [37] addressed the explainability problem by perturbing the importance of training examples and observing their influence on prediction. Similarly, Fong and Veladi [18] also use perturbations to explain predictions. Teso and Kersting [88] recently developed explanations for interactive learners. Though neural networks suffer from lack of interpretability in general, there have been attempts to explain the model through visual analytics, such as Grad-CAM [78] and the more recent work by Zhang and Zhu [100]. However, none of these techniques are applicable to relational data which is the focus of this chapter. Specifically, in relational data there is a single example that is interconnected, and is therefore fundamentally different from the type of data addressed in the aforementioned methods. Related to propositional probabilistic graphical models, more recently, Shih et al. [82] compiled Bayesian networks into a more interpretable decision tree model.

4.2 Query Explanation

Our approach is to extract explanations for a query as a ranked list of MLN formulas, where the ranking encodes the influence of the formula on that query. Before we formally describe our

approach, we motivate it with an illustrative example. Consider a simple MLN with 2 formulas, $f_1 = \mathtt{R}(x) \wedge \mathtt{S}_1(x)$ with weight equal to 0.5 and $f_2 = \mathtt{R}(x) \wedge \mathtt{S}_2(x)$ with weight equal to 0.6. Let $\mathtt{R}$ be the query predicate, and let the domain of x, $\Delta_x = \{X_1, X_2, X_3\}$. Let us assume that we want to explain the results of marginal inference, meaning that we compute the marginal probabilities of $\mathtt{R}(X_1) \ldots \mathtt{R}(X_3)$. Given no evidence, in every possible world, f_2 has a larger influence than f_1 in computing the probability of that world. Therefore, the marginal probabilities of the atoms $\mathtt{R}(X_1) \ldots \mathtt{R}(X_3)$, are influenced more by f_2 as compared to f_1. We illustrate this in Fig. 4.1. Here, we show the exponentiated sum of weights for all satisfied groundings in the first-order formula summed over all possible worlds where the query is satisfied. The values obtained for the formulas f_1 and f_2 are normalized and shown in Fig. 4.1 (a).

However, now consider a second case, where we add evidence $\mathtt{S}_1(X_1)$ and set all other atoms of $\mathtt{S}_1$ and $\mathtt{S}_2$ to false. We now analyze the influence of the formulas in a subset of possible worlds that are consistent with the observed evidence. Here, f_1 now has greater significance than f_2 for the query $\mathtt{R}(X_1)$, since the observed evidence makes the formula f_1 grounded with X_1 true and the formula f_2 grounded with X_1 false. However, when we consider a different query $\mathtt{R}(X_2)$, the influence of f_1 and f_2 changes. Specifically, the influence of f_1 and f_2 on $\mathtt{R}(X_2)$ is equivalent to the case where we had no evidence. This is because case f_1 and f_2 grounded with X_2 have the same truth assignment due to the evidence. Thus, the same set of formulas can have different influences on different queries.

Now, suppose, we add a third formula, $f_3 = \mathtt{S}_1(x) \wedge \mathtt{S}_2(x)$ with weight 0.7. Since, f_3 has the highest weight, we may be tempted to say that f_3 has maximum influence on the probabilities. However, if we quantify the influence of the formulas as before, we get the results shown in Fig. 4.1 (b). Note that adding the formula changes the influence that the other formulas have on the marginal probability. Further, even though f_3 has a higher weight, its influence on the query $\mathtt{R}(X_1)$ is in fact smaller than that of f_2, even in the case where we have no evidence. Thus, we cannot analyze weights of the formulas independently of each other when the atoms are shared among different formulas, since the weights on one formulas can affect the other formulas.

On adding evidence as specified before, the influence of all three formulas are modified as shown. Further, if we assume that the evidence specifies that $\mathtt{S}_1(X_1)$ is true and the other atoms of $\mathtt{S}_1$ and $\mathtt{S}_2$ are unknown (they can be either true or false), then f_3 has a larger influence than the other formulas. Thus, depending upon the evidence as well as the specific query we are looking at, each formula has a different influence on the overall marginal probability. For small examples such as the aforementioned one, we can go over each possible world that is consistent with the evidence and the query, and compute the influence of each formula on the marginal probability of the query. However, this is not practically feasible for large problems. Therefore, we develop a practically feasible solution where we compute the importance based on samples drawn from the distribution over the possible worlds.

To formalize the above example, we first start with some notation. Let $f_1 \ldots f_k$ be the k formulas in the input MLN $\mathcal{M}$. Let $w_1 \ldots w_k$ be weights associated with each of these formulas respectively. Let $\mathbf{Q}$ represent the query predicate, and let $\mathbf{E}$ represent the set of evidence atoms (atoms whose truth assignment is known). Let $q_1 \ldots q_m$ denote the instantiations or ground atoms corresponding to the query predicate. Note that, for the sake of of clarity, we assume that we have a single query predicate, however, it is straightforward to include multiple query predicates.

4.2.1 Sampling

In standard Gibbs sampling for MLNs, we start with a random assignment to all atoms $\omega^{(0)}$ in the MLN except the evidence atoms whose assignments are fixed as given in $\mathbf{E}$. In each iteration of Gibbs sampling, we choose a non-evidence atom based on a proposal distribution α, and compute an assignment to this atom by sampling the assignment based on its conditional distribution. In our case, we assume that α is a uniform distribution, which means that we sample non-evidence atoms randomly in each iteration. From the generated samples, we estimate the marginal probabilities of $P(q_1) \ldots P(q_m)$ as,

$$P(\bar{q}_i) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}(\omega^{(t)} \sim \bar{q}_i) \tag{4.1}$$

where T is the total number of samples, $\omega^{(t)} \sim \bar{q}_i$ denotes that the assignment to atom q_i in $\omega^{(t)}$ is consistent with $\bar{q}_i$. Without loss of generality, we assume that $\bar{q}_i$ refers to the true (or 1) assignment to q_i. Thus, to compute the marginal probability for q_i, we need to compute the ratio of the number of samples where the q_i was equal to true (or 1) and the total number of samples collected.

Suppose we choose to sample a query atom, q_i in an iteration of Gibbs sampling, the main task is to compute the conditional distribution $P(q_i|\omega^{(t-1)} \setminus q_i)$, where $\omega^{(t-1)} \setminus q_i$ is the set of assignments to all atoms except q_i in the sample at iteration $t-1$. Once we compute the conditional distribution, we sample the assignment for q_i, say $\overline{q}_i$ from the distribution, and the subsequent sample $\omega^{(t)} = \omega^{(t-1)} \cup \overline{q}_i$. The conditional distribution to be computed in an iteration is given by,

$$P(\bar{q}_i|\omega^{(t-1)} \setminus q_i) = \exp \sum_j w_j N_j(\omega^{(t-1)} \setminus q_i \cup \bar{q}_i) \tag{4.2}$$

where $N_j(\omega^{(t-1)} \setminus q_i \cup \bar{q}_i)$ is the number of of satisfied groundings in the j-th formula given the assignment $\omega^{(t-1)} \setminus q_i \cup \bar{q}_i$.

We now define the importance distribution for a query atom q_i, $\mathcal{Q}(q_i)$ as follows. In each step of Gibbs sampling, where q_i is satisfied, we measure the contribution of each formula to the Gibbs transition probability. Specifically, for a formula f_k, its contribution to the transition probability is proportional to $\exp(w_j N_j(\omega^{(t-1)} \setminus q_i \cup \bar{q}_i)$, if q_i is the atom being sampled in iteration t. However, since we consider both cases in the conditional probability, namely, the assignment 1 (or true) to $\bar{q}_i$ as well as the assignment 0 (or false) to $\bar{q}_i$, we would like to encode both these into our importance function. To do this, we compute the log odds of a query atom, and score the influence of a formula on the query based on its contribution in computing its log-odds.

Formally, let $\omega^{(t-1)}$ be the Gibbs sample in iteration $t-1$. Suppose we are sampling the query atom q_i, we compute the log-odds ratio between the Gibbs transition probability for $q_i = 0$ and $q_i = 1$. This is given by the following equation,

$$\log \frac{P(q_i = 1|\omega^{(t-1)} \setminus q_i)}{P(q_i = 0|\omega^{(t-1)} \setminus q_i)} = \sum_j w_j N_j(\omega^{(t-1)} \setminus q_i \cup \{q_i = 1\}) - \sum_j w_j N_j(\omega^{(t-1)} \setminus q_i \cup \{q_i = 0\}) \tag{4.3}$$

$$\log \frac{P(q_i = 1|\omega^{(t-1)} \setminus q_i)}{P(q_i = 0|\omega^{(t-1)} \setminus q_i)} = \sum_j w_j (N_j(\omega^{(t-1)} \setminus q_i \cup \{q_i = 1\}) - N_j(\omega^{(t-1)} \setminus q_i \cup \{q_i = 0\}) \tag{4.4}$$

We then update the importance weight of the j-th formula w.r.t query q_i as

$$\mathcal{Q}_j^{(t)}(q_i) \propto w_j N_j(\omega^{(t-1)} \setminus q_i \cup \{q_i = 1\}) - w_j N_j(\omega^{(t-1)} \setminus q_i \cup \{q_i = 0\}) \tag{4.5}$$

We update all the importance weights for q_i denoted by $\mathcal{Q}^{(t)}(q_i) = \mathcal{Q}_1^{(t)}(q_i), \ldots \mathcal{Q}_k^{(t)}(q_i)$ corresponding to the formulas 1 through k in every iteration where q_i is sampled. The importance weight for $\mathcal{Q}_j^{(t)}(q_i)$ after sampling q_i T times is given by,

$$\mathcal{Q}_j(q_i) = \frac{1}{T} \sum_{t=1}^{T} \mathcal{Q}_j^{(t)}(q_i) \tag{4.6}$$

Theorem 4.2.1. *As $T \to \infty$,*

$$log \frac{P(q_i = 1)}{P(q_i = 0)} \propto \sum_j \mathcal{Q}_j(q_i) \tag{4.7}$$

Proof.

$$
\begin{aligned}
log\frac{P(q_i=1)}{P(q_i=0)} = & \\
& \sum_{\omega} \log \frac{P(\omega \sim q_i = 1)}{P(\omega \sim q_i = 0)} \\
& \propto \sum_{\omega} \sum_{j} w_j (N_j(\omega \sim q_i = 1) \\
& - N_j(\omega \sim q_i = 0)) \\
& \propto \sum_{\omega} \sum_{j} w_j (N_j(\omega \sim q_i = 1) \\
& - \sum_{j} N_j(\omega \sim q_i = 0)) \qquad (4.8)
\end{aligned}
$$

where $\omega \sim q_i = 1$ are worlds consistent with the known evidence as well as $q_i = 1$, and $\omega \sim q_i = 0$ are worlds consistent with the known evidence $q_i = 0$. Further

$$\mathbb{E}[\mathcal{Q}_j(q_i)] = \sum_{\omega} w_j (N_j(\omega \sim q_i = 1) - w_j N_j(\omega \sim q_i = 0)) \qquad (4.9)$$

as $T \to \infty$, $\mathcal{Q}_j^{(t)}(q_i) \to \mathbb{E}[\mathcal{Q}_j(q_i)]$, since we are estimating the expectation from worlds consistent with the MLN distribution. Therefore, as $T \to \infty$, $\sum_j \mathcal{Q}_j^{(t)}(q_i)$ $\sum_j \mathbb{E}[\mathcal{Q}_j(q_i)]$ which is equal to the log-odds ratio $log\frac{P(q_i=1)}{P(q_i=0)}$

□

Interestingly, it turns out that in some cases, the importance weights can be obtained without sampling multiple worlds. Specifically, we can show that,

Proposition 1. *If the evidence is complete, i.e., every non-query atom is known to be either true or false, and if every ground formula in the MLN contains exactly one query atom, then* $\mathbb{E}[\mathcal{Q}_j(q_i)]$ $= w_j(N_j(\omega \sim q_i = 1) - w_j N_j(\omega \sim q_i = 0))$*, where* ω *is any world consistent with the known evidence.*

The above proposition implies that, in MLNs where the evidence is fully specified over the

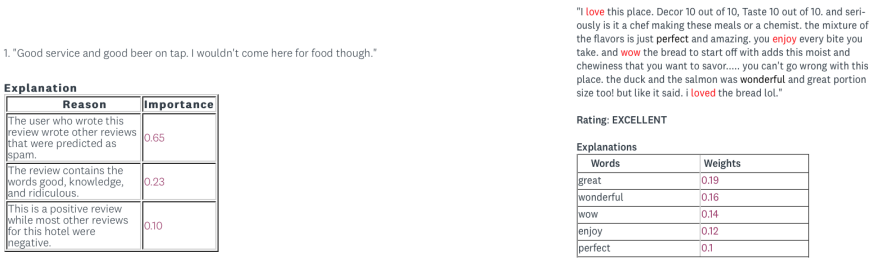
Algorithm 2: Explaining Inference

Input: MLN $\mathcal{M}$, Evidence $\mathbf{E}$, Query atoms $\mathbf{Q}$
Output: Ranking of formulas in $\mathcal{M}$ for each $q_i \in \mathbf{Q}$

```
Initialize the non-evidence atoms in ω^(0) randomly
for t = 1 to T do
    X = Choose a non-evidence atom in ω^(t) uniformly at random
    Flip X in ω^(t) to compute the conditional distribution P(X|ω^(t) \ X)
    Sample X from P(X|ω^(t) \ X)
    if X ∈ Q then
        for each f_j in M do
            Update the importance weight Q_j^(t)(X)
for each q_i ∈ Q do
    Explain q_i as a ranked list of formulas f_1 ... f_k based on importance weights in
      Q(q_i)
```

non-query atoms, and every query atom occurs in an independent subset of ground formulas in the MLN, we can derive the importance weights directly from the specified evidence. However, in cases where the evidence does not cover all the ground atoms, or more than one query atom occurs in a ground formula, we cannot infer its importance without sampling the possible worlds. Note that in general, instead of using Gibbs sampling to generate the possible worlds, we can use Marginal-MAP inference to sum-out the unknown atoms, and then derive the explanations using the evidence. However, marginal-MAP is considerably more expensive [81]. Another strategy is to use the MAP assignment for the unknown atoms. However, this is problematic when we have a significant number of unknown atoms, and if the distribution is multi-modal since, we are essentially considering a single world. A third strategy is to use belief propagation. However, the unknown atoms is again problematic in this case since we need to sum out those atoms to derive the belief propagation messages, and for large number of unknown atoms, this can be extremely expensive. Thus, our sampling strategy allows us to estimate the importance weights in a computationally feasible manner.

Algorithm 2 summarizes our approach. First, we initialize all non-evidence atoms in the MLN randomly. In each iteration, we select a non-evidence atom uniformly at random, and com-



(a) Spam (b) Sentiment

Figure 4.2: Explanations generated by our approach. (a) shows the explanations for the spam prediction application and (b) shows the explanation for the sentiment prediction.

pute the conditional distribution for that atom given the state of all other atoms. Based on this conditional probability, we sample a new assignment for the sampled atom. If the sampled atom is a query atom, for each formula, we compute its importance weight for that query in the current word using Eq. (4.5). We update the importance weight using Eq. (4.5). Once the marginal probabilities in the Gibbs sampler converge, we finally compute a explanation for the marginal probability obtained for each query atom by ranking the formulas in descending order of the importance weights specific to that query.

4.3 Experiments

Our main goal is to evaluate if the explanations output by our approach helps a user understand the "black-box" that is giving this particular explanation. To do this, we designed a comprehensive user study consisting of bout 60 participants. We compared our approach with the explanations given by LIME, an open-source state-of-the-art explanation system. We perform our evaluation using two real-world tasks on a Yelp dataset [64]. We sampled 1000 reviews from this dataset for our experiments. In the first task, we design an MLN that performs joint inference to predict if a review is filtered as a spam review or not by Yelp. In the second task, we predict if a review has positive or negative sentiment based on the review content. We first describe our user study setup and then present the details of our applications along with the results.

4.3.1 User Study Setup

Our user study group consisted of students who have varying backgrounds in Machine learning. The participants were either enrolled in the Machine learning course at University of Memphis or part of the Machine learning club. The participants included undergraduate students, Master's students as well as Ph.D. students. All of them understand classification algorithms and the basics of Machine learning. A few participants were advanced researchers in related areas including Natural Language Processing, computer vision, etc. We divided the participants into two groups, and sent the survey that had the explanations generated by LIME to one group and the explanations generated using our approach to the other group. There were 10 questions in each survey. The first 5 questions asked the participants to rank the explanations on a scale of 1 - 5. The next three questions were used to measure three dimensions of the explanation as follows.

1. Q6: Did the explanations increase your *understanding* of how the classifier is detecting ratings of reviews?

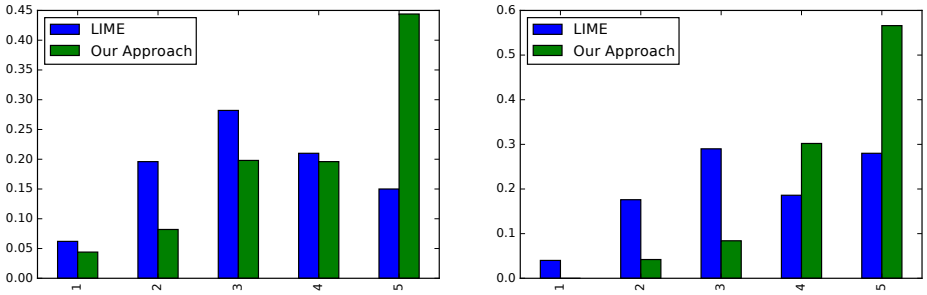
2. Q7: Did the explanations increase your *trust* in the classifier?

3. Q8: Based on the above explanations, will you be able to *apply* this knowledge to predict spam (or sentiment) given a set of new reviews?

Each of the above questions had a response scale of 1 - 5, with 5 being the best score. Finally, we summarized the overall explanation quality by asking participants if they would have liked the classifier to give them more explanations, less explanations or if they felt the explanations provided by the classifier was just right. We also allowed users to enter other comments in free text format.

4.3.2 Application 1: Review Spam Filter

Detecting filtered reviews is a challenging problem. Specifically, unlike say email spam, spam reviews look a lot more authentic since it is designed to influence a user for/against a product/service



(a) (b)

Figure 4.3: Comparison of LIME and our approach using explanation scores as rated by the users. (a) shows this for the spam prediction application and (b) for the sentiment prediction. In each case, we show the $\%$ of users who have given a specific score for an explanation, averaged across all the explanations.

in an open forum. This task more generally called opinion spam has a large body of prior work starting with work by Jindal and Liu [33]. In this case, we develop an MLN that encodes knowledge for detecting spam, and then perform inference on the MLN while generating the explanations.

Our MLN contains formulas that connect words to predicate that indicates whether they are spam $\texttt{Word}(+w, r) \Rightarrow \texttt{Spam}(r)$. We then add relational information into the MLN. Specifically, given two reviews about the same restaurant, the spammer and non-spammer provide ratings that are opposite of each other. For e.g., a spammer provides a positive or high rating, while a non-spammer provides a negative or low rating. Naturally, this is not always true and is therefore a soft formula in the MLN. Finally, we add knowledge that if given two reviews by the same person, if one of them is predicted spam, the other one is likely to be spam as well. In this MLN, note that the evidence variables are the words, we consider the ratings as unknown variables and the query variables are the atoms of predicate `Spam`. Since ratings are unknown, this is a joint inference problem where we infer the rating of a review jointly with inferring if the review is spam or not. We therefore add formulas connecting words with the rating. We learn the MLN by initializing it with weights that we obtain from an SVM [17]. Specifically, we learn an SVM for predicting ratings from the review text, as well as one for predicting if a review is spam/not. Using the coefficients of the MLN, we set initial weights to formulas [17] such as $\texttt{Word}(+w, r) \Rightarrow \texttt{Spam}(r)$, and then use Tuffy [58] to learn the weights of the MLN. The five fold cross validation F1-score

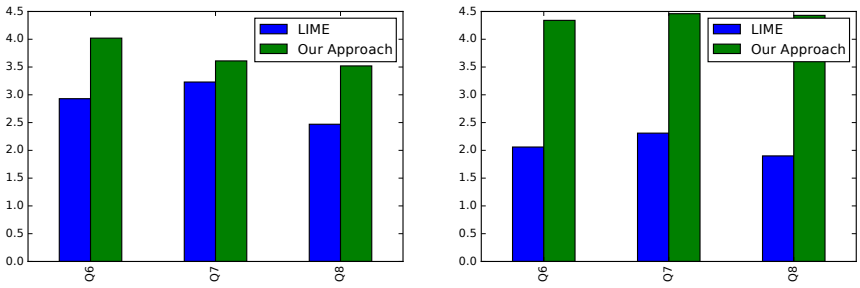
using MLNs for this task was around 0.7. We perform inference and generate explanations for the queries. We picked a small sample of query explanations to conduct the user survey.

Once we perform inference and obtain the importance weights of the formulas, we ranked them, and converted the formula into English to generate the human-readable explanation. We presented the user with this explanation as well as the importance weights (normalized) for the 5 most important formulas. An example of the explanation generated is shown in Fig. 4.2 (a). The users could look at the original review and rate the explanation for that review. For LIME, we provided the input which is the review content and since LIME does not explain relational information, it uses the non-relational features (words/phrases) to come up with its explanation using SVMs as the base classifier.

The comparison of the user response scores for the explanations is shown in Fig. 4.3. As seen here, on average, across the reviews in the survey, a larger percentage of users gave our explanations higher scores as compared to the explanation generated by LIME. On the other hand, a large percentage of users rated LIME explanations around the halfway mark (score 3). Further, when we analyze the responses over the three explanation dimensions as shown in Fig. 4.4 (a), we see that our approach was favored by participants in all three dimensions. Particularly, the dimensions of understanding the classifier and being able to use the knowledge in the explanation scored much higher. This shows that including higher-level relational knowledge in the explanations makes the explanations richer and more appealing to humans.

4.3.3 Application 2: Review Sentiment Prediction

In this application, we predict if a review has has a positive or negative sentiment based on the words in that review. Specifically, we have MLN formulas that connect words in the review to the sentiment. However, we assume incomplete evidence. That is, we remove a small set of words from the review and therefore, their state is unknown. The inference task is to jointly infer the state of the hidden words along with predicting if it is a positive or negative sentiment review. To do this, we add relational knowledge to the MLN. Specifically, we encode MLN formulas that a user

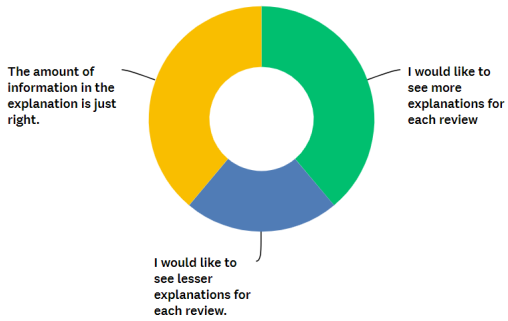


(a) Spam (b) Sentiment

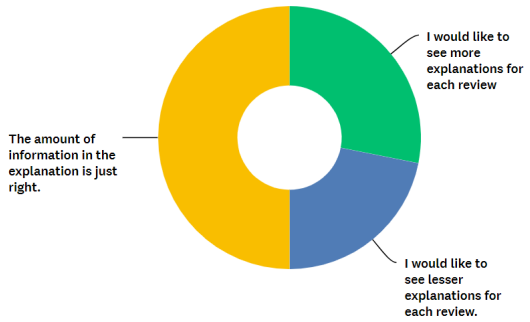
Figure 4.4: Comparison for the average scores given by users for 3 key dimensions related to the explanations. Q6 measures understanding of the classifier, Q7 the trust in the classifier and Q8 if they can replicate the classifier based on the explanation. Higher scores are better. (a) shows results for spam prediction and (b) shows results for sentiment prediction

is likely to use the same words to describe a positive or negative rating. Thus, we can use words from other reviews written by the same user to predict the sentiment of a review. We learn the MLN using a similar procedure as described in the previous section. Our five-fold cross validation accuracy here was around 0.9.

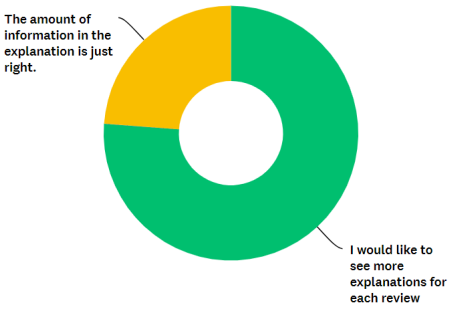
In this case, we generate explanations in terms of word formulas only. Specifically, for each review, we explain its predicted sentiment as a set of words (and their corresponding importance weights). Note that these words can contain missing words (inferred to be true) as well as words known to be true (due to evidence). Thus, LIME and our approach generates the same form of explanations (words and weights) as shown in Fig 4.2 (b). However, since we can infer the states of hidden words, our explanation is richer than that generated by LIME. Fig. 4.3 (b) shows the comparison of the explanation scores for LIME and our approach. Here, we see a very similar trend to the results for the spam prediction application. Specifically, most users though that our approach yields very good applications, while LIME explanations was considered average. Further, Fig. 4.4 (b) illustrates that our approach was significantly better in terms of helping users understand, trust and apply the prediction method. This shows that using relational knowledge can yield a more comprehensive explanation (in the presence of noisy/unknown variables).



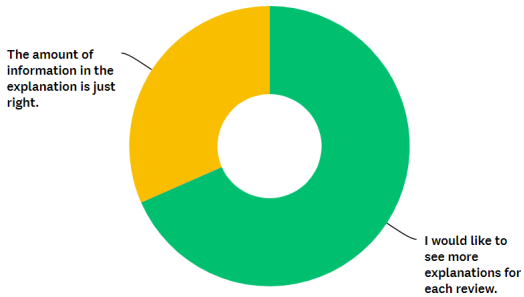
(a) Our Approach (Spam)



(b) Our Approach (Sentiment)



(c) LIME (Spam)



(d) LIME (Sentiment)

Figure 4.5: Comparison of user responses for .the question that summarizes the effectiveness of explanations. (a) and (b) show this for spam prediction and sentiment prediction using our approach, and (c), (d) show the responses for LIME.

4.3.4 T-Test

We use the T-Test to compare the means of the two user groups (those who evaluated LIME and our approach). The null hypothesis for the t-test is that there is no difference between the means of the two groups. In our case, it will mean that our explanation is no better or worse than the lime explanation. The alternate hypothesis is that the means of the two groups are not the same, in which case it will mean that our explanation is either better or worse than the lime explanation. We performed the t-test on the responses to the summary question regarding the quality of the explanations. We coded these as follows. i) rating for lime explanation (coded as group 1) ii) rating for our explanation (coded as group 2). The response options are, i) would like to see more explanation (coded as 1) ii) would like to see less explanation (coded as 2) and iii) Right amount of explanation (coded as 3). The coding is based on the desirability of the response. We assumed

the best case is the right amount of explanation and therefore coded this as the highest. Then, we assumed that requiring less amount of information is worse than right amount of information, and is therefore coded as 2. Finally, we assumed that a user requiring more amount of information is the worst case (coded as 1) because our main motivation is to make the explanation human-interpretable. Thus, according to our coding, the higher mean will be considered better because we coded the right amount of information as the highest. We obtained $p = 0.03$ (< 0.05). Therefore, the difference in explanations provided by our approach is statistically significant. Thus, we can reject the null hypothesis and our explanation is at least better or worse than LIME explanation.

The mean and standard deviations of the two approaches (based on the coding) is as follows. *LIME explanation has a mean of 1.63 and a standard deviation of 0.95. The explanations based on our approach has a mean of 2.22 and a standard deviation of 0.87.* Therefore, our explanations are clearly preferred by the users as compared to the explanations given by LIME. The full breakdown of the responses is shown in Fig. 4.5. As seen here, in each of the two tasks, users considered our explanations to be better than LIME. Interestingly, even in the case where the type of explanations was identical (words explaining the ratings), LIME produced worse results than our approach (see Fig. 4.5 (b) and (d)) which takes advantage of relational dependencies across different reviews.

4.4 Summary

Explanations of predictions made by machine learning algorithms is critical in several application domains. In general, MLNs are interpretable models models but it is challenging to explain results obtained from inference over MLNs. In this chapter, we presented an approach where we explain the results of relational inference in MLNs as a ranked list of formulas that encode their influence on the inference results. Specifically, we compute the importance weights of the MLN formulas based on how much they influence the transition probabilities of a Gibbs sampler that performs inference in the MLN. The importance function is specific to the query and changes as we vary the evidence. We conducted a comprehensive user study to evaluate the effectiveness of

our explanations and compared our explanations with LIME, a state-of-the-art explanation method for non-relational data. We explained the results of two real-world prediction problems, namely, predicting spam reviews and predicting sentiment of a review from text. On both these problems, we showed that the explanations generated by our approach was much more human-interpretable as compared to the explanations generated by LIME.

Never-the-less, still it is challenging issue to extract the reliable explanations from MLNs when its size increases significantly. Further, it is required to make the explanations rich and more comprehensive to users. We have developed a novel approach to address these issues in the next chapter.

Chapter 5

Interpretable Explanations for Probabilistic Inference in Markov Logic

In this chapter, we develop an approach for human-interpretable explanations in statistical relational models (SRMs). Specifically, we explain probabilistic inference in Markov Logic Networks (MLNs) [15], a widely used SRM.

MLNs are a symbolic AI model that represent uncertain relational knowledge using weighted first-order logic formulas. The MLN represents a joint distribution as a probabilistic graphical model, where each potential in the graphical model corresponds to a symbolic formula in the MLN. Thus, a natural approach to explain inference in MLNs is to quantify the influence of formulas in the overall joint distribution. However, weights attached to the formulas do not have a direct probabilistic interpretation. Therefore, explaining the influence of individual formulas in probabilistic inference is not straightforward. To do this exactly, it turns out that we need to compute the partition function of the distribution which is computationally infeasible in practice.

To explain inference in MLNs, we formalize explanations as expected values of formula states. We can then estimate these expectations from samples drawn from the distribution of the MLN using approaches such as Markov Chain Monte Carlo. However, while this approach yields sound explanations, in practice, explaining large MLNs is a hard problem. Specifically, the underlying probabilistic model becomes very large and well-known sampling methods such as Gibbs sampling have poor *mixing* properties in such large MLNs [74]. Consequently, explanations derived from a poorly mixed sampler tend to produce results that have poor interpretability. Note that this is analogous to explanations in non-relational classifiers as well, where complex decision boundaries are less explainable. Therefore, well-known approaches such as LIME [66] and

SHAP [48] provide explanations by approximating complex decision boundaries with simpler, surrogate models. In a similar spirit, here, we derive explanations from an approximate MLN that improves the interpretability of the explanation. Specifically, we construct *coalitions* of formulas where a coalition tries to maintain the relational structure present in the original MLN. To do this, we leverage symmetries in the MLN, i.e., variables that have similar relational structure to other variables in the MLN. We derive coalitions with a reduced number of variables by sampling from groups of (approximately) symmetric variables in the MLN. We then explain probabilistic inference outcomes in the coalitions.

However, once we have multiple explanations from different coalitions, it turns out that combining these explanations is challenging in our case since it is hard to weight the explanations. Ideally, the weights should encode the distance between the coalition to the true distribution, i.e., give closer approximations of the MLN a larger weight in the overall explanation. However, computing the true distribution in our case is intractable. Therefore, we develop a weighting method that penalizes explanations where the influence of formulas significantly deviates from the MLN parameterization. That is, if the ranking of formulas in the explanation tend to match the rankings based on the MLN weights, then the explanation is more consistent with the MLN parameterization and therefore is given a larger weight. We integrate the explanations by combining the weighted ranking orders in each explanation. Specifically, the optimal global explanation is one that maintains the weighted ranking order in explanations over all coalitions. We use a sampling based approach [42] to solve this hard combinatorial problem that converges to the optimal ranking asymptotically.

We show through experiments using annotated explanations that our approach can focus on relevant explanations better than other approaches such as LIME, SHAP or relational explainers that generate explanations using the full MLN distribution [16]. Further, we also conduct a user study to evaluate the effectiveness of our explanations from a human user perspective.

5.1 Related Work

Guidotti et al. [25] survey different explanation approaches in Machine learning and organize them as outcome explanation methods or model explanation methods. Outcome explanations are used for "black-box" classifiers that are complex to explain. A common theme here is to explain the black-box classifier with a simpler model. LIME [68] explains an instance by perturbing it and learning a local, simple decision boundary to classify the perturbed instances. SHAP [47] generates coalitions of features and quantifies the influence of a feature by minimizing the loss between a simple classifier on the coalitions and the original model. Koh and Liang [37] use influence functions from robust statistics to measure the change in training parameters due to a small change in the data. Fong and Veladi [18] use interpretable image perturbations to recognize salient image features. Suderrajan et al. [87] developed integrated gradients as an approach to explain deep networks more generally. More recently, Shao et al. [80] developed an approach using influence functions to correct the model during training such that it gives better quality explanations.

In symbolic AI models, Darwiche and Hirth [10] developed a formal framework for explaining classifier decisions using ordered binary decision diagrams compiled from a Bayesian network. Shih et al. [82] explained Bayesian networks by compiling them into decision trees. Roy et al. [72] developed explanations for activity recognition in videos using tractable probabilistic models with inputs from deep learning methods. Farabi et al. [16] explained MLN formulas but unlike our approach they generate explanations using the full MLN which can be non-interpretable if the MLNs represent large, complex distributions. More recently, Broeck et al. [4] analyzed the complexity of SHAP and showed that it is intractable even for simple distributions.

5.2 Interpretable Explanations

We motivate our approach with a simple example. Consider an MLN with three formulas, $\texttt{Smokes}(x) \Rightarrow \texttt{Asthma}(x)$; $\texttt{Smokes}(y) \wedge \texttt{Friends}(x, y) \Rightarrow \texttt{Asthma}(x)$ and $\texttt{Smokes}(x) \Rightarrow \texttt{Smokes}(x)$. Generally speaking, the first formula is a good explanation for an individual having asthma. However,

for an individual, say $Alice$ who does not smoke, $\texttt{Asthma}(Alice)$ is explainable if a lot of Alice's friends are smokers. Thus, the relational dependencies between $Alice$ and other individuals is an effective explainer for $Alice$ having asthma. However, suppose $Alice$ is friends with a large number of individuals, it becomes harder to quantify the influence of each of these individuals since their smoking habits depend upon other individuals who are friends with them. In general, as the domain (number of individuals or objects) become larger, the probabilistic graphical model underlying the MLN becomes complex and extracting the important dependencies for a specific query becomes harder.

A common strategy that is used in well-known explanation methods for black-box classifiers is to estimate the hard-to-interpret black-box classifier boundary using simpler *surrogate* functions. For instance, SHAP creates *feature coalitions* that combines subsets of features and learns a surrogate model from these coalitions from which the explanation is derived. Similarly, LIME perturbs instances in the neighborhood of the predicted instance and learns a surrogate model for explaining the classification of the perturbed instances in the local-neighborhood. Following the same principle, here, we simplify the MLN creating smaller coalitions of formulas. We rank the importance of formulas within each coalition and then combine these explanations together. Since each of the explanations are extracted from a simpler MLN, the inference results on the model are more reliable and the explanations for the inference results are likely to be more interpretable.

5.2.1 Explanation Framework

To formalize our explanation framework, we begin with some notation. Let $\mathcal{M}$ represent the MLN and $P_{\mathcal{M}}$ represent the distribution of the MLN. Let $\mathbf{f}(Q)$ represent the set of ground formulas containing the query atom Q and let $\mathbf{E}$ represent the set of evidence atoms. Note that here, we assume that each atom is either an evidence or a query atom. Let w_f represent the weight of a ground formula f (note that all groundings of a first-order formula share the same weight in the regular MLN semantics).

Definition 1. *The explanation for query Q in $\mathcal{M}$ with evidence $\mathbf{E}$ is denoted by $\sigma(Q)$ is a permu-*

tation π over $\mathbf{f}(Q)$.

Given an importance weighting function $I()$, where $I(f, Q)$ indicates the importance of ground formula f on determining the probability of the query $P(Q)$, a sound explanation is defined as follows.

Definition 2. *$\sigma(Q)$ is a sound explanation if π orders $f_i \in \mathbf{f}(Q)$ such that $I(f_{\pi_i}, Q) \geq I(f_{\pi_{i+1}}, Q)$.*

In the remainder of the chapter, when we refer to an explanation given an importance function, we assume that it is a sound explanation. To define $I()$, an intuitive approach is to directly use the MLN parameters. Specifically, $I(f, Q) = w_f$, where w_f is the weight of f. However, this simple approach is problematic since the importance of a formula changes dynamically based on the observed variables $\mathbf{E}$. For example, let $\texttt{Fever}(x) \Rightarrow \texttt{Flu}(x)$ have a larger weight compared to $\texttt{Cough}(x) \Rightarrow \texttt{Flu}(x)$. However, given evidence that an individual has cough and not fever, we can explain that the individual has flu using the second formula. Similarly for a different query, the first formula may be more important than the second. Thus, the underlying problem with assuming that the MLN weights determine the importance of a formula is that the parameters do not correspond to probabilities. For the explanation to be meaningful, given evidence $\mathbf{E}$, $I()$ should encode the influence of a formula on the conditional probability $P(Q_1, \ldots Q_n|\mathbf{E})$, (note that we drop the subscript $\mathcal{M}$ from the distribution to make it more readable) where $Q_1, \ldots Q_n$ represent the query atoms. We define this as follows.

Definition 3. *Let $\mathbb{E}[n_f]$ be the expected truth value of formula f w.r.t the joint conditional distribution $P(Q_1, \ldots Q_n|\mathbf{E})$ and $\mathbb{E}_Q[n_f]$ be the expected truth value of f w.r.t the marginal distribution of Q, i.e., $P(Q|\mathbf{E})$, we define the importance weight of formula f for a query Q as*

$$I(f, Q) = \mathbb{E}_Q[n_f] - \mathbb{E}[n_f] \tag{5.1}$$

Intuitively, using the above definition, $I(f, Q)$ has a larger value if f is true in more worlds where Q is also true and false in more worlds where Q is false. More specifically, we relate the importance $I(f, Q)$ to the partial derivative of the marginal probability w.r.t to the weight of f.

Proposition 2. $I(f,Q) = \frac{\partial \log(P(Q|\mathbf{E}))}{\partial w_f}|$.

Proof. Let $\mathbf{Q}_{-Q}$ represent all the query atoms other than Q. The log marginal probability is computed by summing out all the query atoms in $\mathbf{Q}_{-Q}$. Therefore we have,

$$P(Q|\mathbf{E}) = \sum_{Q' \in \mathbf{Q}_{-Q}} \frac{1}{Z} \exp(\sum_f w_f n_f) \tag{5.2}$$

$$logP(Q|\mathbf{E}) = log \sum_{Q' \in \mathbf{Q}_{-Q}} \exp(\sum_f w_f n_f) - log(Z) \tag{5.3}$$

$$\tag{5.4}$$

Z is the normalization constant given by

$$\sum_{Q,Q' \in \mathbf{Q}_{-Q}} \exp(\sum_f w_f n_f)$$

.

Let,

$$Z' = \sum_{Q' \in \mathbf{Q}_{-Q}} \exp(\sum_f w_f n_f)$$

Taking partial derivatives, we have,

$$\frac{\partial logP(Q|\mathbf{E})}{\partial w_f} = \frac{1}{Z'}\frac{\partial Z'}{\partial w_f} - \frac{1}{Z}\frac{\partial Z}{\partial w_f}$$

Simplifying the terms, we get,

$$\sum_{Q' \in \mathbf{Q}_{-Q}} P(Q,Q'|\mathbf{E}) n_f - \sum_{\bar{Q},Q' \in \mathbf{Q}_{-Q}} P(\bar{Q},Q'|\mathbf{E}) n_f$$

where $\bar{Q}$ denotes a possible assignment to Q. Thus, we have,

$$\frac{\partial logP(Q|\mathbf{E})}{\partial w_f} = \mathbb{E}_Q[n_f] - \mathbb{E}[n_f]$$

□

Thus, from the above theorem, the importance $I(f, Q)$ is proportional to the influence of f on $P(Q|\mathbf{E})$. Note that the importance of a formula is dynamic (as compared to the MLN weight) since it depends on the query as well as the evidence.

5.2.2 Sampling-based Importance Estimation

From Eq. (5.1), to compute $I(f, Q)$, we need to compute expectations w.r.t $P(Q|\mathbf{E})$ and $P(Q_1, \ldots Q_n|\mathbf{E})$. Therefore, computing $I(f, Q)$ is computationally infeasible since computing the expectations is equivalent to solving the marginal inference problem which is well-known to be in $\#P$. Therefore, we use a sampling-based approach to approximate $I(f, Q)$.

Specifically, we use Gibbs sampling (though in theory other samplers can be utilized) due to its efficiency to estimate $I(f, Q)$. Specifically, we generate samples from the distribution and approximate the expected values $\mathbb{E}_Q[n_f]$ and $\mathbb{E}[n_f]$. To implement this, we start with a random configuration of $\mathbf{Q}$. In each iteration, we sample a random query Q from its conditional distribution $P(Q|\mathbf{Q}_{-Q})$. If the sampled value is $Q = 1$ (or `True`), then for all `True` formulas that contain Q, we update the estimate for $\mathbb{E}_Q[n_f]$ and for all `True` formulas, we update the estimate for $\mathbb{E}[n_f]$.

Algorithm 1 summarizes our sampling-based approach for explanations. Based on the convergence of the probabilities for the Gibbs sampler in Algorithm 1, clearly, the expected values in Eq. (5.1) converge to the true expected values. Therefore, as $T \to \infty$, $\bar{I}(f, Q) \to I(f, Q)$.

5.2.3 Influence of Domain-Size

The *mixing time* of the Gibbs sampler in Algorithm 1 is the time that the sampler takes to reach the stationary distribution, namely, the distribution $P_{\mathcal{M}}$. $I(f, Q)$ is estimated from samples after the sampler reaches its stationary distribution, i.e., after a period called its *burn-in* time. If $I(f, Q)$ is estimated from samples before burn-in, the explanations will be unreliable since they are not consistent with the distribution represented by the MLN. However, the main issue is that for large

Algorithm 3: Explanations

Input: MLN $\mathcal{M}$, evidence $\mathbf{E}$, queries $\mathbf{Q}$
Output: Explanations for $\mathbf{Q}$
//
$\bar{\mathbf{Q}}$ = random assignment to $\mathbf{Q}$
for $t = 1$ *to* T **do**
  for *Each atom* $Q \in \mathbf{Q}$ **do**
    Sample Q from $P(Q|Q_{-Q}, \mathbf{E})$ and update $\bar{\mathbf{Q}}$
    for *each* f *containing* Q **do**
      // update sufficient statistics for $\mathbb{E}[n_f]$
      Update the count $n_1(f, Q)$ if f is true
      // update sufficient statistics for $\mathbb{E}_Q[n_f]$
      if Q *is sampled as true* **then**
        Update the count for $n_2(f, Q)$ if f is true
  for *each* Q *in* $\mathbf{Q}$ **do**
    for *each* f *containing* Q **do**
      $\bar{I}(f, Q) = \frac{1}{t}\, n_2(f, Q) - n_1(f, Q)$

MLNs, the mixing time can be extremely large and consequently, the explanations generated from the samples are likely to be of non-interpretable.

Intuitively, when the MLN structure is more complex, the distribution becomes harder to sample and explain. This can be formalized by results in Sa et al. [74]. Specifically, the mixing time depends upon the *total influence* in the MLN distribution. To define this, let $\mathbf{B}_j$ denote the set of all pairs of states ($\bar{X}$, $\bar{Y}$). Each each state is an assignment to all atoms, and each pair ($\bar{X}$, $\bar{Y}$) $\in \mathbf{B}_j$ differ in an assignment to a single variable j. The total influence is the maximum total variational distance between distributions obtained by conditioning an atom on the assignments in the pairs defined in $\mathbf{B}_j$. Formally,

$$\alpha = \max_{V \in \mathcal{M}} \sum_{j} \max_{(\bar{X}, \bar{Y}) \in \mathbf{B}_j} ||P(V|\bar{X}_{-V}) - P(V|\bar{Y}_{-V})||_{TV} \tag{5.5}$$

where $P(V|\bar{X}_{-V})$ is the conditional distribution of an atom $V \in \mathcal{M}$ given assignments to all the other atoms.

For larger values of α in Eq.(5.5), in [74] it is shown that the Gibbs sampler takes longer

to mix. Thus, using Algorithm 1, it becomes harder to generate meaningful explanations. Specifically, in our case, the difference between $P(V|\bar{X}_{-V})$ and $P(V|\bar{Y}_{-V})$ is dependent upon the influence of a single atom. Specifically, if the change to an atom's assignment modifies the truth values of several ground formulas, then the total variational distance in Eq. (5.5) increases.

Formally, let $C(f, R)$ represent the variables in formula f that are part of predicate R. Let $C(f, R)^-$ represent variables in formula f that are not part of predicate R. Let $(\bar{X}, \bar{Y}) \in \mathbf{B}_j$ and j denote an atom of predicate type R. For any atom V, the variational distance between $P(V|\bar{X}_{-V})$ and $P(V|\bar{Y}_{-V})$ is proportional to the number of groundings of f whose truth value (either 0/1) differ in the two distributions. The atom j occurs in $\prod_{v' \in C(f,R)^-} |\Delta_{v'}|$ ground formulas, where $\Delta_{v'}$ is the domain of variable v'. Therefore, a change to the state of j can potentially change the truth value of all these ground formulas. Thus, as the domains of the variables increase, the variational distance between the state pairs in $\mathbf{B}_j$ increases and the total influence becomes larger. Algorithm 1 therefore generates non-interpretable explanations in MLNs with large domains. To generate interpretable explanations, we simplify the MLN such that the total influence in the simplified MLN is smaller than that in the original MLN.

5.2.4 Relational Coalitions

To reduce the total influence, we create *coalitions* of ground formulas that are much smaller than the full MLN. On such coalitions, the sampler in Algorithm 1 mixes faster and is more likely to generate interpretable explanations. In principle, this approach is similar to LIME and SHAP where the original classifier is approximated by a simpler, more explainable classifier. One way to generate coalitions is to sample ground formulas in $\mathbf{f}(Q)$ randomly for each query Q. However, since the MLN is a relational model a randomly sampled coalition may not preserve the relational dependencies present in the original MLN. To illustrate this, consider the graph underlying an MLN shown in Fig. 5.1.Suppose we randomly sample this graph, we may end up with coalitions shown in Fig. 5.1 (b). However, this does not truly capture all dependencies in the original graph. Instead, if we sample the graph to obtain coalitions shown in Fig. 5.1 (c), though this model is simpler,

we still retain the dependencies specified in the original graph. More generally, by exploiting symmetries in the MLN, we create coalitions that preserve relational dependencies.

Definition 4. *Given evidence* $\mathbf{E}$*,* X_1 *and* X_2 *are exchangeable objects if* X_1 *and* X_2 *can be exchanged in the ground formulas such that the formulas which were satisfied (or unsatisfied) by* $\mathbf{E}$ *before the exchange remain satisfied (or unsatisfied) after the exchange.*

Two coalitions $\mathbf{L}_1$ and $\mathbf{L}_2$ are symmetric if for every formula in $\mathbf{L}_1$ can be uniquely mapped to a formula in $\mathbf{L}_2$ such that the objects in $\mathbf{L}_1$ are exchangeable with objects in $\mathbf{L}_2$. For example, the coalition, $\mathtt{R}(X_1) \wedge \mathtt{S}(X_1, Y_1)$; $\mathtt{R}(X_1) \wedge \mathtt{S}(X_1, Y2)$ is symmetric to $\mathtt{R}(X_2) \wedge \mathtt{S}(X_2, Y_1)$; $\mathtt{R}(X_2) \wedge \mathtt{S}(X_2, Y2)$ if X_1 is exchangeable with X_2. Suppose we are given equivalent coalitions, $\mathbf{L}_1$ and $\mathbf{L}_2$, where a ground formula where $f \in \mathbf{L}_1$ is mapped to $f' \in \mathbf{L}_2$, then, if f is an explanation to a query w.r.t $\mathbf{L}_1$, we project f' as the explanation to Q w.r.t $\mathbf{L}_2$.

While truly exchangeable objects may be rare in practice, we can soften the constraints in our definition to allow for approximately exchangeable objects. Specifically, we define a continuous approximation for the exchangeability between X_1 and X_2, $\delta(X_1, X_2)$ based on the number of ground formulas whose assignments differ before and after the exchange of X_1 and X_2. However, computing this for large MLNs is hard and it is shown in prior work [95] that counting the satisfied groundings of a formula given evidence is a computationally hard problem. Therefore, it is infeasible to exactly compute $\delta(X_1, X_2)$. However, we instead leverage scalable learning methods to learn a dense vector representation that approximately encodes how similar X_1 is to X_2. Specifically, as in [29], we learn embeddings over objects in the MLN using an approach commonly used in word embedding methods. Specifically, suppose a ground formula f that is satisfied by the evidence contains objects $(X_1 \ldots X_k)$, we predict X_i from $X_1 \ldots X_{i-1}, X_{i+1} \ldots X_k$. Thus, if X and X' are predicted from similar objects (or have the same context), this means that X and X' can be exchanged without significantly altering the truth values of the formulas in which they occur. To learn the embedding, we use existing implementations of skip-gram models [51]. Note that several other graph-based methods such such as those computing automorphisms in the MLN graphs to recognize symmetries [5, 56], locality-sensitive hashing [55], etc. can be used as well. However, it

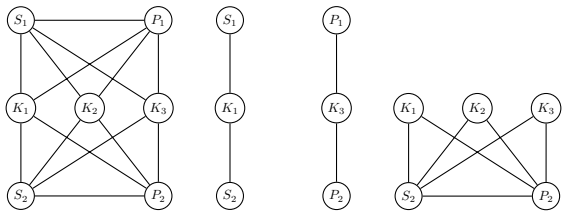


Figure 5.1: (a) Original MLN Graph. (b) Simplification by random sampling. (c) Simplification that preserves relational structure.

is easy to see that, to compute the embedding in our case, we do not explicitly construct the MLN graph which can be very large, and therefore we can scale up even to large MLNs.

Given the embeddings, we cluster the objects to generate coalitions such that for a query, we choose formulas in the coalition that can effectively substitute for formulas that are not chosen in the coalition.

Definition 5. *Given a clustering of objects based on their embeddings, if f is a formula with objects ($O_1 \ldots O_k$), $\theta(f)$ is an approximately symmetric formula where each O_i is substituted by O_i' that is in the same cluster as O_i.*

We construct a coalition by selecting formulas based on a clustering of objects. Specifically, we sample objects from each cluster such that the number of objects sampled is proportional to the cluster size. Let $\mathbf{O} = O_1 \ldots O_m$ be the sampled objects. For every query atom that can be formed from the sampled objects, say Q, we construct the coalition $\mathbf{f}'(Q)$ as follows. Initially, we start with an empty $\mathbf{f}'(Q)$. For each ground formula f that contains Q, we include f in $\mathbf{f}'(Q)$ if we cannot find $\theta(f) \in \mathbf{f}'(Q)$. Thus, we reduce the original set of formulas $\mathbf{f}(Q)$ to $\mathbf{f}'(Q)$ such that for $f \in \mathbf{f}(Q) \setminus \mathbf{f}'(Q)$, there exists a $\theta(f) \in \mathbf{f}'(Q)$. This means that, suppose the optimal explanation for Q is a ground formula f, then the coalition can potentially generate f or $\theta(f)$ as its explanation.

5.2.5 Integrating Multiple Explanations

By combining coalitions across all sampled queries, we generate an MLN and jointly explain all the sampled queries using Algorithm 1. However, note that since the embeddings and the clustering

is approximate, using a single coalition, we may not obtain the formulas needed to explain a query effectively. Therefore, we generate multiple coalitions and explain each independently. We then integrate the explanations generated across all the coalitions.

Let $\sigma_i(Q)$ represent the explanation obtained for query Q using $\mathcal{M}_i$ which is the MLN generated in the i-th iteration of sampling the clusters. Note that since we are sampling the clusters, not every query will be a part of each of the generated MLNs. Therefore, if a query $Q \notin \mathcal{M}_i$ we can generate an explanation for Q using $Q' \in \mathcal{M}_i$ that is in the neighborhood of Q in the embedding assuming that the two explanations are symmetric. Specifically, if f is a formula in the explanation for Q', we explain Q with a substitution $\theta(f)$ that contains Q.

Given explanations $\sigma_1(Q) \ldots \sigma_m(Q)$, where each explanation is from a coalition that tries to cover the full relational structure of the MLN, we now generate a unified explanation for Q. Specifically, we want to know the influence of a formula f on a query across the coalitions. In the case of non-relational models, note that predictions are typically easy to perform. Therefore, for each coalition, it is relatively simple to optimize the loss between the prediction by the original model for that coalition instance with the predication made by the approximate, simpler model (e.g. a linear model) to derive the final explanation across coalitions. However, in the case of relational models, inference is hard and therefore, we cannot assume that we can perform inference in the original model in the first place. That is, generating results based on the full MLN is a hard problem when the MLN is large even using approximate inference methods.Therefore, instead of optimizing the loss between inference results from the original MLN with inference results from the coalitions, we derive weighted explanations, where the weight approximately encodes the difference between the original MLN and the simpler MLN. The weighted explanations are then combined into a unified global explanation for the model.

Coalition Weighting

Let β_i denote the weight for the explanations derived from $\mathcal{M}_i$. Note that, ideally, we want to weight $\sigma_i(Q)$ based on the distance between $\mathcal{M}_i$ and $\mathcal{M}$. However, this is infeasible since the exact

distribution of $\mathcal{M}$ is intractable to compute. Therefore, we instead weight each coalition based on how the importance weights computed from the coalition match with the MLN parameterization. Influence functions proposed in [37] use a similar approach where the change in model parameters relative to perturbed parameters is used as a way to quantify the effect of the perturbation.

Formally, let $\mathcal{M}$ be the original MLN and $\mathcal{M}_i$ be the MLN that was generated from the coalitions. Let $\mathbf{w}$ be the weights (or parameters) of $\mathcal{M}$ and let $\mathbf{w}_i^*$ be the optimal parameterization for $\mathcal{M}_i$. This means, if $\mathbf{w}_i^*$ was used to parameterize $\mathcal{M}_i$, then the importance weights for explanations from the MLN would ideally match with its formula weights. Suppose we are explaining query Q_j using $\mathcal{M}_i$, we compute the importance weights according to Eq. (5.1). In Proposition 1, we show that the importance weights computed for Q_j is equivalent to the gradient vector for the likelihood function of Q_j. Suppose this gradient has a large norm, this means that to the importance weights are significantly different from the weights $\mathbf{w}$. That is a large gradient norm implies that $\mathbf{w}$ must be significantly changed to reach $\mathbf{w}_i^*$. Thus, the explanations generated by $\mathcal{M}_i$ have a larger bias and should therefore be weighted down relative to the other generated simple MLNs. On the other hand, if the importance weights imply a small gradient norm, then the difference between $\mathbf{w}$ and $\mathbf{w}_i^*$ is small.

In general, we can now weight the coalitions based on the importance weights of the formulas generated while explaining the queries in the coalition. That is, if the importance weights of the formulas show a large variation over all the queries, then those coalitions will have a smaller weight compared to coalitions where the importance weights have small variation over all the queries. Specifically, let $\mathbf{I}_i$ represent the matrix of importance weights obtained from $\mathcal{M}_i$, where the j-th row corresponds to weights for query Q_j. Let $z_i = ||\mathbf{I}_i||_F$, where $||\mathbf{I}_i||_F$ denotes the Frobenius norm for the weight matrix. Let $\bar{Z} = \sum_i z_i$. We weight the explanations from $\mathcal{M}_i$ with $\beta_i = 1 - \frac{z_i}{\bar{Z}}$.

Unified Explanation Ranking

Let $S(Q) = \cup_i \sigma_i(Q)$, i.e., the union of explanations from all MLNs. To make equations more readable, we drop the Q since it is implicit that explanations are specific to a query. Let τ represent an ordered (or ranked) subset of k explanations, i.e., $\tau \subset S$. We now define a distance function between τ and each of the explanations, where the explanation for $\mathcal{M}_i$ is weighted by β_i as,

$$\Phi(\tau) = \sum_i \beta_i d(\tau, \sigma_i) \tag{5.6}$$

$$d(\tau, \sigma_i) = \sum_{t \in \tau \cup \sigma_i} |R_i(t) - R_\tau(t)|$$

where $R_i(t)$ is t's ranking in σ_i and $R_\tau(t)$ is its ranking in τ. Note that if t is not in τ, we set $R_\tau(t)$ to the maximum value $k+1$, and similarly if t is not in σ_i, we set $R_i(t)$ to $k+1$. We formulate the optimal explanation as,

$$\tau^* = \arg\min_\tau \Phi(\tau)$$

Solving the optimization problem for τ^* exactly is computationally hard since we need to enumerate all possible subsets of size k. Therefore, we use an approximate approach to combine the weighted explanations using the Cross-Entropy Monte Carlo (CEMC) [42] algorithm. Specifically, let $\mathbf{v}$ be a $n \times k$ probability matrix, where n is the total number of formulas in the union of all explanations and k is the size of the global explanation that we seek to find. Each column in $\mathbf{v}$ represents a multinomial distribution over the formulas. To obtain a global explanation, we can draw a sample from the distribution $P_\mathbf{v}$ to generate $\mathbf{x}$ such that each column in $\mathbf{x}$ has exactly a single 1 and each row sums to at most 1. Given that $\mathbf{v}$ is the current set of parameters, in [42], it is shown that new parameters $\mathbf{v}'$ that minimizes the KL-divergence between the current probability distribution and the ideal distribution is given by maximizing,

$$\mathbb{E}_{\mathbf{v}}[\mathcal{I}(\Phi(f(\mathbf{x},\mathbf{v})) \leq y) log P_{\mathbf{v}'}(\mathbf{x})]$$

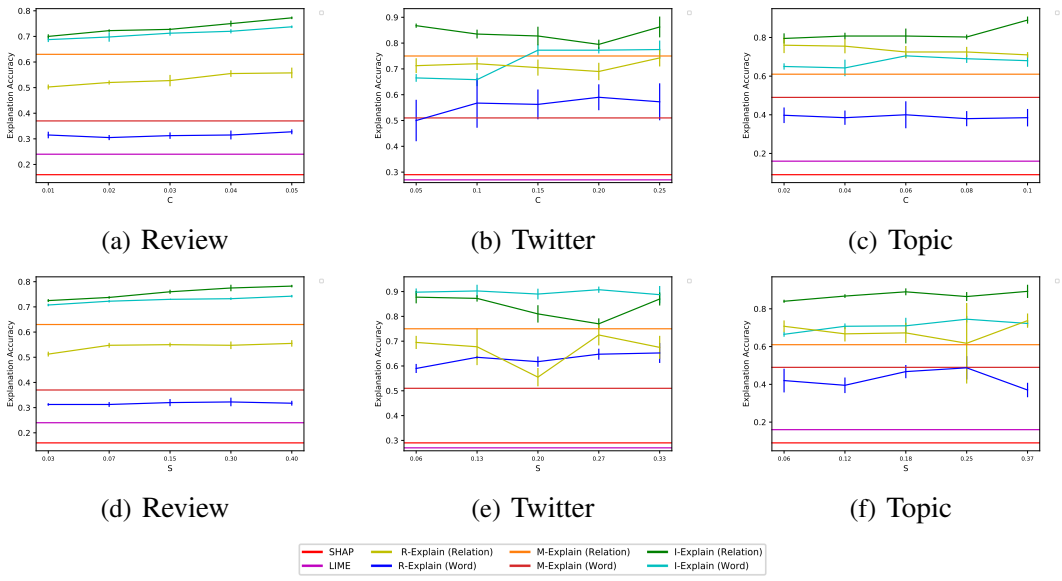
where $\mathcal{I}$ is an indicator function and $f(\mathbf{x},\mathbf{v})$ denotes a τ that has been drawn from $P_{\mathbf{v}}$. To do this, we draw samples from $P_{\mathbf{v}}$ and count the proportion of samples for which the objective function value is smaller than a given y. Specifically,

$$v_{new} = \frac{\sum_{i=1}^{N} \mathcal{I}(\Phi(\tau_i) \leq y)\mathbf{x}_i}{\sum_{i=1}^{N} \mathcal{I}(\Phi(\tau_i) \leq y)} \tag{5.7}$$

CEMC learns an approximation for the optimal global explanation as follows. We initialize the probability matrix $\mathbf{v}_o$ as a uniform distribution for each column. In each iteration, we consider N samples, where each sample is a possible explanation. Half the samples are drawn from $\mathbf{v}_i$ and we augment it with half of the best samples (best objective function values) from $\mathbf{v}_{i-1}$. We then order the explanations in ascending order of their objective values computed using Eq. (5.6). Suppose $\Phi_1 \ldots \Phi_N$ are the ordered objective values for the explanations, we choose y to be the ρ-quantile (for a suitable rho) objective value. Using this, we obtain the updated $\mathbf{v}_{i+1}$ from Eq. (5.7). After the parameters in the probability matrix converge, we output the final explanation as the one that corresponds to the best objective function value, i.e., $y = \Phi_1$. CEMC produces explanations that are asymptotically unbiased. That is, suppose the values of Φ_1 is the sequence y_1, y_2 ..., this converges to the optimal objective value $y*$ at y_∞.

5.3 Experiments

We evaluate our approach along three dimensions, i) we evaluate accuracy of explanations based on manual annotations, ii) we evaluate the importance of information present within the explanations and iii) we evaluate the usability of explanations through a user-study.



(a) Review (b) Twitter (c) Topic

(d) Review (e) Twitter (f) Topic

Figure 5.2: Comparing explanation accuracy for varying values of C ans S. The mean values for 10 runs are plotted along with error bars that indicate Std-Dev.

5.3.1 Data and Tasks

We use three datasets in our evaluation. The first dataset is a dataset sampled from Yelp [64] for review sentiment classification that contains 1544 reviews. Our second dataset consists of 650 COVID tweets from Kaggle. We classify whether a COVID-19 tweet contains useful information or not. We manually annotated tweets based on whether they contain facts that can be considered as useful information (e.g. scientific details, policies, etc.) or if the tweets are non-informative. Our third dataset uses a portion of the well-known 20newsgroups dataset from the UCI repository. Here, we classify articles as automative related articles or otherwise. We had a balanced dataset of 1000 articles. The MLNs in each case consist of word formulas, i.e., connecting words with the query and relational formulas that encode homophily. That is, if two queries are linked then they share the same class. In reviews, the links are defined by reviews written by same user and those about the same restaurant. In the twitter and topics data, the links are defined by tweets (or topics) written by the same user.

5.3.2 Implementation

We refer to our approach as `I-Explain`. To learn the embeddings for MLN objects, we use open source code from [29]. We learn the MLN weights using a hybrid approach since the word formula weights are hard to learn directly. Specifically, we initialize the word formula weights using SVM coefficients and then learn the relational formula weights conditioned on the SVM weights as described in [17]. We used an open source R package for CEMC (with default parameters) to combine the coalition explanations. We applied `LIME` and `SHAP` to our tasks using the word features. We also developed a baseline where we created coalitions by randomly sampling formulas (denoted by `R-Explain`. Finally, we also compared our approach with explanations from the complete MLN as described in [16] which we refer to as `M-Explain`.

We manually annotated the ground truth for the explanations. Specifically, since all our tasks are text based, for each instance, we pick words that best explain the class. To avoid bias, these were annotated by two people independently and the final annotation was the common explanations chosen by both. Annotating relational formulas manually is hard, i.e., it is hard to judge which relational formulas are good explanations to a query. Therefore, we use a different annotation method here. Specifically, note that in all our MLNs, every relational formula connects exactly two query atoms. Therefore, for a query Q, for each true grounding of the relational formula f where Q occurs, we compare the similarity of Q with Q', where Q' is the other query that occurs in f. We compute the document vectors for Q and Q' (using Gensim Doc2Vec) and measure the similarity between these vectors. If queries Q, Q' have similar content and the relational formula where Q and Q' occurs is true, then, the formula is annotated as an explanation for both Q and Q'.

5.3.3 Explanation Accuracy

We compared the explanation accuracy of `I-Explain` with `LIME`, `SHAP`, `R`-Explain and `M-Explain`. We computed the accuracy as the $\%$ of matches of the generated explanations with the annotated explanations for each dataset. For `I-Explain`, `R-Explain` and `M-Explain`, we used the top 5 ranked word formulas and the top 5 ranked relational formulas as the explanation. Note that for

Dataset	Method	Top-Exp-Acc (F1)	Peak-Acc (F1)	Std-Dev
Review Classification	I-Explain	0.82	0.84	0.01
	R-Explain	0.54	0.58	0.04
	M-Explain	0.6	0.68	0.01
	SHAP	0.53	0.58	0.01
	LIME	0.55	0.61	0.02
Tweet Classification	I-Explain	0.79	0.83	0.01
	R-Explain	0.61	0.77	0.08
	M-Explain	0.6	0.69	0.01
	SHAP	0.33	0.44	0.01
	LIME	0.46	0.54	0.01
Topic Classification	I-Explain	0.7	0.75	0.008
	R-Explain	0.58	0.66	0.04
	M-Explain	0.62	0.67	0.01
	SHAP	0.39	0.54	0.02
	LIME	0.53	0.56	0.01

Table 5.1: Prediction Accuracy (F1-score) using only the explanation (Top-Exp-Acc) and the peak accuracy obtained when we add the formulas (or features) in ranked order according to explanation importance (Peak-Acc). The values shown are mean values for 10 runs and Std-Dev is the average Std-Dev of the Top-Exp-Acc and Peak-Acc.

LIME, SHAP, we only obtain word explanations and therefore only measure accuracy on these. We evaluated our approach by varying two parameters C and S, C controls the number of clusters and S controls the number of samples drawn from each cluster to create the coalitions. For a domain-size of $|\Delta_x|$, we use $C * |\Delta_x|$ clusters and if a cluster contains N instances, we use $S * N$ samples from that cluster. For R−Explain, we pick random formulas to create coalitions such that it matches the number of formulas in $\mathtt{I} - \mathtt{Explain}$ for a fair comparison. We ran the experiments 10 times and report the mean and variance of the accuracy. Our results are shown in Fig. 5.2, where we independently show the accuracy on word explanations and relational formula explanations. For M−Explain, LIME and SHAP, clusters don't play any role, so their accuracy is constant. Fig. 5.2 (a), (c), (e) keeps S constant at 5%, while (b), (d), (f) keeps C constant at 2% for the review and topic data, and 5% for twitter (since it is around half the size) and varies S. From the results, we see that I−Explain consistently shows better accuracy than other methods. As C increases, we have larger coalitions and the performance is fairly stable with slightly increasing accuracy but can also dip in some cases (see Fig. 5.2 (c)). This indicates that with larger coalitions, we may have more uncertainty (since the expected values are estimates), therefore explaining smaller coalitions is an advantage in relational models.

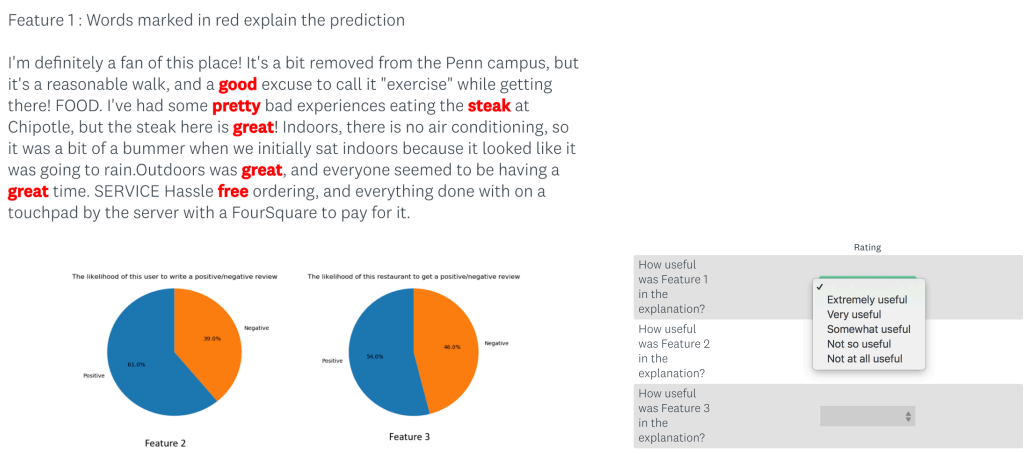
5.3.4 Explanation Information Content

Next, we evaluate the importance of explanations based on the information that they provide to the prediction model. This is similar to the idea in Performance Information Curves [34] that has been recently proposed for XAI in computer vision tasks. Here, we determine the accuracy of the model when it only uses the explanations. If explanations contain content that is more important in the model, then, they should be able to make accurate predictions using only the explanations. Table 1 summarizes the accuracy in terms of F1-score when we only add the explanations for each query and use them to make predictions. Further, we also show the peak accuracy that is achieved by the model as we add formulas (or features in the case of LIME or SHAP) in order of explanation importance. We show the results in Table 5.1 where we indicate the mean F1-score for the predicted queries for 10 runs as well as the standard deviation. As seen here, `I-Explain` has better accuracy when we only use the explanation for the prediction. Also, the peak accuracy is close to the accuracy using explanations in most cases which validates that explanations indeed represent highly informative content.

5.3.5 Usability

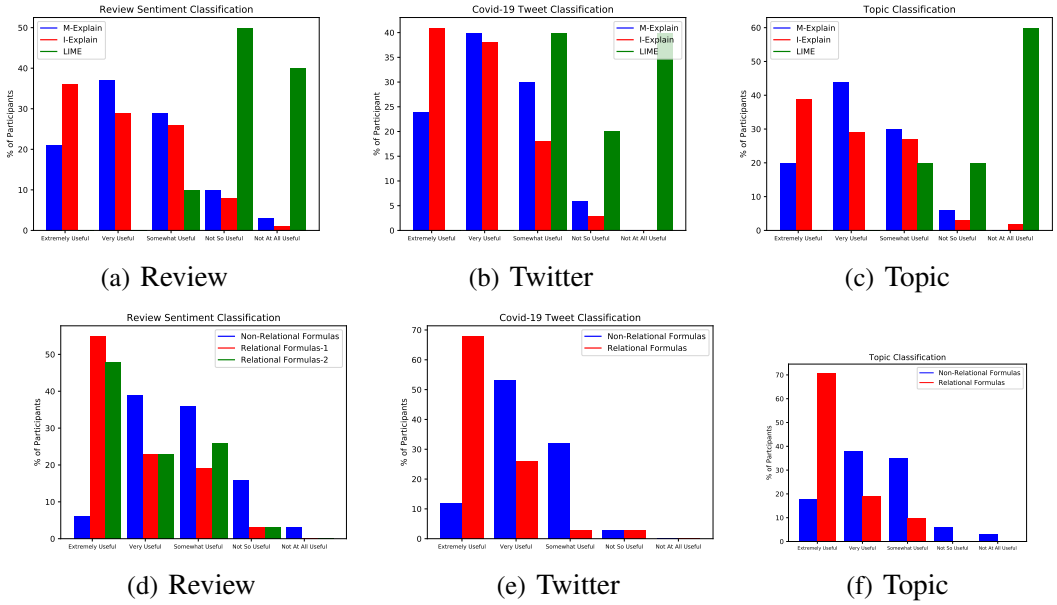
We evaluate the usability of explanations through a user study. Our goal here is to understand if users find the explanations useful. We recruited 50 graduate students in this study who were currently (or formerly) enrolled in a Machine Learning course. Note that this demographic is likely to constitute most likely XAI users since one of the key applications of XAI is to help experts debug Machine learning methods. We divided the overall group into 3 sub-groups and a student was given explanations from just one of the methods (which was not named) to avoid bias. Specifically, a user given explanations from two different methods may be biased towards one that simply shows him/her more information. To ensure sufficient responses for each method, we used `I-Explain`, `M-Explain` and `LIME` (since `SHAP` explanations are similar in format to `LIME`) in this study.

Explanation Dashboard. We present our explanations in the form of dashboard shown in Fig. 5.3.



(a) (b)

Figure 5.3: Explanation Dashboard and survey questions.



(a) Review (b) Twitter (c) Topic

(d) Review (e) Twitter (f) Topic

Figure 5.4: (a) - (c) The y-axis shows the % of participants who rated the explanation feature with the rating specified in the x-axis where the % is normalized by the number of features in the explanation. (d) - (f) User ratings for relational and non-relational features for `I-Explain`.

The first feature presents the word formulas marked in the data. The remaining features correspond to relational formulas. Visualizing relational formula explanations is not as straightforward since relationships are not explicitly seen in the data (as opposed to words). To visualize this, we average the importance weights of the relational formulas in the explanation and display this as a graph (normalized between 0 and 1) indicating support of the relational formulas in the prediction. Our survey consisted of 12 randomly chosen explanations where we used 4 explanations for each task (2 of them corresponding to each class). For every explanation, we asked users to rate the usefulness of each feature in the dashboard as shown in Fig. 5.3 (b) on a Likert scale.

Results. The results of the survey are shown in Fig. 5.4. Fig. 5.4 (a) - (c) show the % of user responses for each value in the Likert scale normalized by the number of features in the dashboard. We see here that `I-Explain` and `M-Explain` had better scores than `LIME`. This shows that overall, users preferred to see explanations with both relational and non-relational formulas. Further, users also preferred `I-Explain` to `M-Explain` since the explanation quality was better due to the use of simplified models. That is, `M-Explain` tends to provide poor explanations when the MLN is large. Fig. 5.4 (d) - (f) further shows the breakdown of scores for `I-Explain`. Interestingly, across all datasets while users felt both types of features are important, a larger percentage found relational features to be extremely useful in the explanations compared to non-relational features. The mean user score for the relational features in the explanation was 4.42 while for the non-relational features, it was 3.84. A two-sided t-test showed that the difference in user responses for relational and non-relational features was statistically significant.

5.4 Summary

Explaining the results of marginal probabilistic inference in an interpretable manner is hard when MLNs have large domains. We presented an approach that constructs simplified models from the MLN to generate more interpretable explanations. The simplified explanations are then weighted and combined into a unified explanation. Our results on several problems illustrated that our approach generates high quality explanations for relational data.

Chapter 6

Improving Explanations using Feedback

AI and Machine learning (ML) has had unprecedented successes over the last decade in almost all application domains. The use of AI/ML for decision making in critical areas such as healthcare, criminal justice and defense has accelerated the need for explainable AI (XAI) [26]. Over the last few years, tremendous progress has been made in XAI with the development of several different methods that perform model explanations and/or outcome explanations [25] for machine learning methods. Specifically, model explanations explain the behavior of the model as a whole in an human-interpretable format (e.g. decision trees). However, several powerful machine learning models may simply be too complex to explain and are thus "black-boxes" to a user. Fortunately though, even in such complex models, we can explain individual predictions made by these models. Specifically, outcome explanations explain reasons for individual predictions made by a "black-box" model. Arguably, two of the most well-known general-purpose outcome explanation methods include Locally Interpretable Model-Agnostic Explanations (LIME) [67] and SHAP [48] based on Shapley values. Both of these approaches rank the importance of features by using a simpler interpretable model that approximates the original black-box classifier. However, these approaches explain instances independently, i.e., they do not consider relationships that can exist between instances which are ubiquitous in the real-world. For example, reviews written by the same user are related to each other, people with similar social circles can have shared interests, etc. Therefore, in this chapter, we develop an approach to generate richer outcome explanations using relationships in the data.

We represent relational knowledge using the semantics of Markov Logic Networks (MLNs).

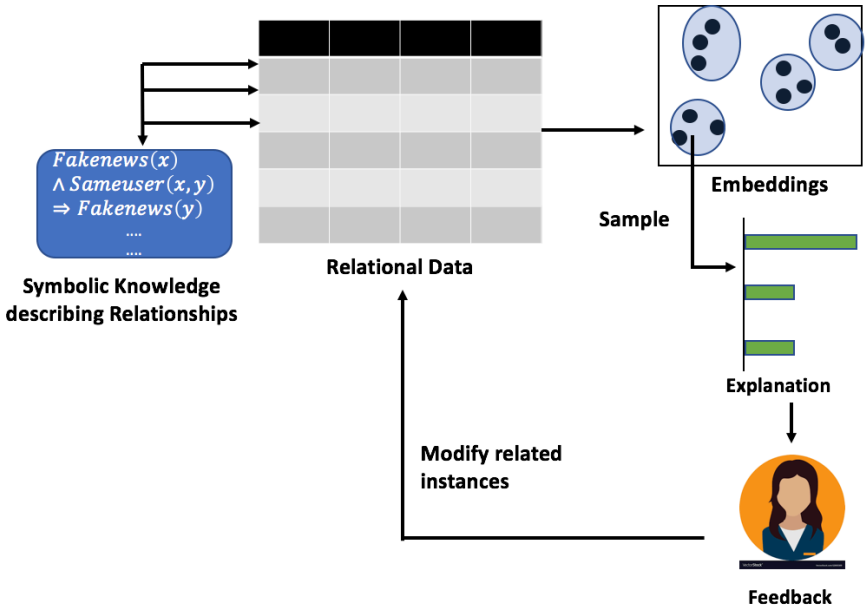


Figure 6.1: A schematic illustration of our approach.

In MLNs, relationships are described in the form of first-order logic (FOL) formulas. MLN formulas represent *soft constraints* since they are not always true (active) or false (inactive). Thus, the formulas have a degree of uncertainty as opposed to pure FOL formulas. We then add relationships specified in the MLN into outcome explainers in the form of embeddings. Specifically, similar embeddings for instances imply that the MLN specifies similar relationships for these instances. Using these embeddings, we generate explanations from LIME and SHAP for classification tasks. Next, since explanations are inherently subjective, it is sometimes hard to arrive at the "optimal" explanation without the perspective of a human. Therefore, we add a human-in-the-loop to provide feedback to explanations. We obtain feedback for a small set explanations and then propagate this feedback to instances based on the symmetries between their respective embeddings. That is, we assume that for similar/symmetrical instances using similar feedback will help improve their explanations. Using this feedback, we modify the MLN formulas by sampling active formulas based on how likely they are to generate explanations that are consistent with the feedback. We then re-learn the embeddings from on the modified MLN to generate more interpretable explanations.

An illustration of this framework is shown in Fig. 6.1.

We evaluate the performance of our approach in three text classification problems. Specifically, we compare the performance of LIME and SHAP using several standard classifiers. Further, we also develop a comparative approach using BERT which is an attention-based mechanism in deep networks [91]. We evaluate the approaches on i) the quality of explanations generated when compared to human annotated explanations, ii) the relevance of the content in the explanation to the classifier and iii) the usability of relational knowledge in explanations based on a user study.

6.1 Related Work

There has been significant progress in XAI over the last few years. Guidotti et al. [25] survey a number of explanation methods and come up with a taxonomy of XAI methods. They broadly classify XAI methods as model explanations and outcome or prediction explanations. In model explanations, the goal is to explain the behavior of the model as a whole. In outcome explanations, the goal is to explain individual predictions even though the model itself may be too complex as a whole. In [6], the authors provide an overview of the reasons, domains and the foundational definitions for explanations in supervised Machine learning algorithms. In [43], the author suggests that explanations are required whenever the AI model cannot explain what the user of the model needs to know. Explainability has also been extensively studied in various other disciplines such as psychology and cognitive sciences. In [54], the authors provide an overview of these perspectives. The influence of social sciences on explainablility in AI is studied in [52].

More specific to prediction explanations which are also called post-hoc explanations, the goal is to explain predictions even though the full model may be very complex to explain. For example, some models are naturally transparent such as decision trees while others such as neural networks are not. Explanations for predictions made by black-box classifiers typically take a similar general approach where the hard to explain classifier is approximated by a more interpretable model [54]. According to [43], post-hoc explanations are natural language based, visualizations, interactive in nature or based on approximations or case-based reasoning. LIME

developed by Ribeiro et. al. [68] is one of the most popular outcome explainers. LIME is based on approximating the local decision boundary of a classifier with an interpretable boundary. Lundberg and Lee [47] developed another popular outcome explainer called Shapely Additive Explanations. Here, the attributionsto features towards the prediction made by the classifier is quantified based on Shapley values that was originally used in economics. Ross et al. [70] developed an explanation approach as a regularizer to simplify a classifier's decision boundary. Koh and Liang [37] developed perutbation-based models where we perturb the data to observe changes to the classifier. Fong and Veladi [18] also use a similar perturbation-based approach to explain predictions. Due to the popularity of deep neural networks, several approaches have specifically focused on explaining deep network prediction. For instance, there have been attempts to explain neural networks through visual analytics, such as Grad-CAM [78] and Zhang and Zhu's approach [100]. Suderrajan et al. [87] developed a popular approach called integrated gradients to explain deep network predictions more generally based on the gradient values. For the case of relational models, there have been relatively fewer general explanation methods. In particular, Farabi et al. [16] proposed an approach using Markov Logic Networks (MLNs) that provide explanations in relational data.

6.1.1 Feedback-based Models

Feedback has also been used to correct or modify features particularly under the umbrella of active learning. For example, DUALIST [79] is a tool that was developed for incorporating feedback in an active learning framework. DUALIST incoporates feedback on the instances (the user labels the specific class of an instance) or at the feature-level to reduce dimensionality (selecting features appropriate for a class). Several other approaches based on active learning [101, 44] also use feedback to improve the learning model.

There have also been previous explainable AI approaches that try to interact with users and take their feedback. Teso and Kersting [88] developed explanations for interactive learning methods. Recently, Ghai et al. [22] developed an explanation interface for active learning. The idea is to develop explanations for interactive learning where the model can selectively query a machine

teacher who explains decisions taken by the model. Smith et al. [86] performed an analysis of explanations. In particular, they studied whether revealing explanations for the ML method motivated users to fix the ML method. Further, they showed that ability to provide feedback impacted perceptions of the user regarding the ML method. Importantly, they also showed that generating explanations without the ability to give feedback caused frustrations to the users of the ML model. This validates our approach here to provide users the ability to provide feedback and take corrective actions for the explanations based on the feedback. Kulesza [40] et al. proposed an explanation debugging interface. Just like debugging a program, the machine learning model could be modified by users through interactions. Their results showed that by simply interacting with such a system, participants could gain a deeper understanding of the model. Thus, in general, it has been shown in multiple studies that feedback combined with explanations significantly contributes to an increase of user's trust in the ML algorithm. Our approach is similar in this regard since we use feedback to generate more meaningful explanations for a classifier making the classifier's predictions more interpretable.

6.2 Relational Explanations

Typically ML classifiers assume non-relational data. That is, the instances in the data are assumed to be independent and identically distributed (i.i.d). Therefore, explainers for these classifiers explain the prediction of an instance in the data independently from the others. Specifically, they rank the relative importance of features for a particular instance in the data. However, in many real-world cases, we have relationships among different instances in the data. For example, in the case of twitter data, the tweets written by the same user are related to each other. Here, we assume that there are relationships in the data and these relationships are defined in a symbolic language. Specifically, we assume that the relationships in the data are defined using Markov Logic Networks (MLNs). Note that it is certainly possible to define the relationships using other languages such as knowledge graphs [32] or probabilistic logic programs [11]. Here, we use MLNs as our language of choice since it has intuitive semantics and is rich enough to represent complex knowledge since

it is based on first-order logic. We use MLNs to define formulas based on our understanding of the domain and then inject knowledge from these formulas into outcome explainers. Thus, the explanation generated by the explainer is augmented with relational information from the domain.

6.2.1 Representing Relational Knowledge

An MLN consists of first-order logic (FOL) formulas, where the formulas have a symbolic and graphical meaning. Specifically, we can *ground* the formulas with objects/constants to obtain a ground formula and this represents a relationships among all atoms in that formula. For example, the FOL formula, $\texttt{FakeNews}(x) \wedge \texttt{SameUser}(x, y) \Rightarrow \texttt{FakeNews}(y)$ represents a FOL formula that connects two articles written by the same user. Specifically, it encodes our prior knowledge that if two articles are written by the same user, if one them is a fake article, then the other one is fake as well. A specific instantiation of the formula grounds the formula with specific objects, i.e., articles in this case, such as $\texttt{FakeNews}(X_1) \wedge \texttt{SameUser}(X_1, Y_1) \Rightarrow \texttt{FakeNews}(Y_1)$. Thus, we have a relationship defined over the articles X_1 and Y_1. This relationship is a logical connection between atoms $\texttt{FakeNews}(X_1)$, $\texttt{SameUser}(X_1, Y_1)$ and $\texttt{FakeNews}(Y_1)$. The same logical relationship can also be represented in a graph where the nodes are the three atoms and a clique between the nodes represents the relationship specified by the symbolic formula. Thus, given a domain with a large number of objects, we have a large implicit graph that encodes several relationships among the objects in that domain. We define two types of formulas, *relational* and *non-relational* formulas. The non-relational formulas represent features for a specific instance and the relational formulas represent logical connections between instances. For example, a formula such as $\texttt{Word}(W, X_1) \Rightarrow \texttt{FakeNews}(X_1)$ is a formula that encodes that the object (or feature) W (word in this case) is present in instance X_1. An example of a relational formula is $\texttt{FakeNews}(X_1), \wedge \texttt{SameUser}(X_1, Y_1) \Rightarrow \texttt{FakeNews}(Y_1)$ encodes the logical relationship between the instances X_1 and Y_1. For ease of exposition, we assume that a relational formula connects exactly two instances, though in general a formula can logically connect any number of instances. The overall end goal is to classify all instances. We use the term *query objects* interchangeably with instances since these are the objects

that need to be classified and are distinct from objects that represent features (e.g. words in the above example). We begin with some definitions to formalize our representation.

Definition 6. *An* active *ground formula is one that is logically satisfied by the data.*

Based on the above definition, it is easy to see that active formulas represent the possible relationships between objects as defined in the data. Note that the same objects can be involved in many ground formulas, some of which may be active and some inactive depending upon the data. For example, if a dataset asserts that $\texttt{FakeNews}(X_1)$, $\texttt{SameUser}(X_1, Y_1)$ and $\texttt{FakeNews}(Y_1)$ are true, then from the FOL formula defined in our previous example, $f_1 = \texttt{FakeNews}(X_1), \wedge \texttt{SameUser}(X_1, Y_1) \Rightarrow \texttt{FakeNews}(Y_1)$ is an active formula, but $f_2 = \texttt{FakeNews}(X_1), \wedge \texttt{SameUser}(X_1, Y_2) \Rightarrow \texttt{FakeNews}(Y_2)$ is not active.

Definition 7. *The context of an object X for an active formula f denoted by $\mathcal{C}(X, f)$ is the set of all objects that X occurs with in f.*

$\mathcal{C}(X, f)$ denotes the set of all objects that X is related to in f. From our running example, since f_1 is active, X_1 and Y_1 are in each other's context w.r.t f_1 but not w.r.t f_2.

Definition 8. *X_1 is symmetric to X_2 in formula f denoted by $S(X_1, X_2, f)$ if X_1 occurs in f and there is a formula f' where X_2 occurs and $\mathcal{C}(f, X_1) = \mathcal{C}(f', X_2)$.*

Definition 9. *X_1 is symmetric to X_2 denoted by $S(X_1, X_2)$ if $S(X_1, X_2, f) \; \forall \; f$ that X_1 occurs in.*

The definition of symmetry between objects can also be softened to allow for approximate symmetries. Specifically, if instead of all formulas, suppose for a large number of formulas, $f_1 \ldots f_k$, we have $S(X_1, X_2, f_i)$, then, we can be reasonably confident that X_1 and X_2 share similar relationships based on the data. Thus, $\hat{S}(X_1, X_2)$ denotes the approximate symmetry between X_1 and X_2 where in a reasonably large number of formulas, the two objects are symmetric. Now, given these definitions, we need to learn a representation for the objects such that approximately symmetric objects share a similar representation. To do this, we use an approach previously developed [29] that learns object representations using word embedding approaches.

6.2.2 Embeddings

Consider a graphical representation of an MLN shown in Fig. 6.2. The similar colors indicate instances which have the same features, i.e., the non-relational formulas are equivalent across these instances. The connections indicate the relational formulas where active formulas are shown by a solid line and inactive ones are shown by a dashed line. Our goal is to learn embeddings for instances such that if they have similar neighborhoods (only considering active formulas), their embeddings are close to each other. Thus, in the example, we will have two clusters of embeddings $\{X_4, X_6\}$ and $\{X_1, X_3, X_7, X_9\}$. Note that for the objects that represent features we also learn embeddings in an analogous manner, where if a feature occurs with similar features in several instances, they have the same embedding. For example, in the case of text features, words have similar embeddings when they appear in similar contexts (i.e., their neighboring words are similar) across several instances.

Let $\mathbf{X}$ represent the set of objects. Given the training data, for every active formula f, we predict all objects in $\mathcal{C}(X, f)$ from X. To do this, we use existing skip-gram embedding architectures such as Word2Vec [51]. Specifically, the objective is as follows.

$$\max \sum_{X \in \mathbf{X}} \sum_{f} \log P(\mathcal{C}(X, f)|X)$$

From [51], we maximize the above objective using a neural network. Specifically, the training data consists of a one-hot encoded input object X and we predict the one-hot representation of $X' \in \mathcal{C}(X, f)$. The hidden layer in the neural network encodes the representation for the objects. Specifically, recall that if $\hat{S}(X, \hat{X})$, over a large number of formulas, the context of X and $\hat{X}$ will be similar to each other. Therefore, X and $\hat{X}$ will predict similar objects and to do this, the hidden layer will learn to assign similar representations for X and $\hat{X}$. Specifically, if v_X is the vector representation for X and $v_{X'}$ is the vector for X', then, the probability $P(X'|X) \propto \exp(v_X^\top v_{X'})$. Thus, the probability of predicting one object from another is proportional to the similarity between their vectors. Therefore, we have the following result.

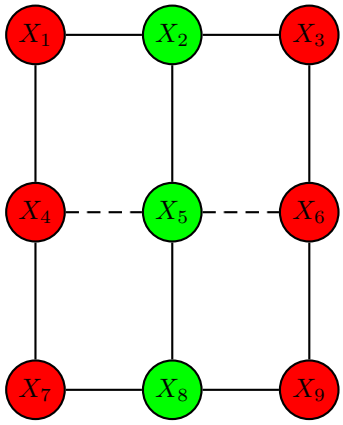


Figure 6.2: An example showing the graphical representation of an MLN, where the same colored nodes mean that the features in those instances are similar to each other. An active relational formula is shown by a solid line. The embeddings are similar if two nodes are connected to similar nodes in their active formulas. Thus, $\{X_4, X_6\}$ and $\{X_1, X_3, X_7, X_9\}$ form two groups of similar embeddings.

Proposition 3. *Given objects X_i, X_j if $S(X_i, X_j)$, then $v_{X_i} = v_{X_j}$.*

Proof. $S(X_1, X_2)$ indicates that the contexts of X_i and X_j are identical over all formulas. Thus, in training the model, for both X_i and X_j, we make predictions over the same set of objects. Let Y_1 ... Y_k be this set of objects. The objective of the skip-gram model is to maximize the log-likelihood of predicting a context object, i.e.,

$$\sum_k logP(Y_k|X_i) + logP(Y_k|X_j)$$

Based on the vectors assigned to the objects, to maximize the above log-likelihood we need to maximize the term $v_{X_i}^\top \sum_k v_{Y_k} + v_{X_j}^\top \sum_k v_{Y_k} = (v_{X_i} + v_{X_j})^\top \sum_k v_{Y_k}$. Let $\sum_k v_{Y_k} = v_s$. The maximization term can then be written as $(v_{X_i} + v_{X_j})^\top v_s$. The optimal solution to this maximization is when $v_{X_i} = v_{X_j}$. □

In computing the embeddings, a subtle detail is that we assumed the active formulas to be known apriori. However, this may not always be the case. For example, if we have a formula such as $\texttt{FakeNews}(X_1), \wedge \texttt{SameUser}(X_1, Y_1) \wedge \texttt{FakeNews}(Y_1)$. To know if this formula is active, we need to know if the objects X_1 and Y_1 are classified as fake news. Thus, in general, if the query

is part of a formula, then we cannot assume that we know the truth value of the formula. One common assumption is the closed world assumption, where unknown truth values are assumed to be false. In our case, we use a modified approach where we fill in the missing labels of the queries. Specifically, we train a separate classifier to classify the query objects and assign a label to all the query objects before computing the embeddings. These labels therefore act as a prior to compute the embeddings. Thus, the truth values for each formula is completely known since all the queries have an assigned label, and we then construct the embeddings from the active formulas.

6.2.3 Explanations

Note that if X_i and X_j are approximately similar, i.e., $\hat{S}(X_i, X_j)$, we obtain a continuous approximation of this similarity based on the distance between the vectors v_{X_i} and v_{X_j}. We use this similarity to explain predictions on the query objects.

Let $\mathbf{X}$ represent the set of *query* objects. We explain the prediction $f(X)$ for $X \in \mathbf{X}$ made by a classifier $f()$ as a ranked list of relational and non-relational formulas signifying their importance in the prediction. We next describe generating these explanations using LIME and SHAP.

LIME Explanation

Let X be the query object for which we generate explanations. Let $\bar{x}$ represent the set of formulas containing X. If a formula can influence the prediction of X, i.e., if it is an active formula, we assign a 1 corresponding to that formula in the binary vector. We now sample set of binary vectors $\mathbf{Z}$ in the neighborhood of the binary vector $\bar{x}$. Thus, $Z \in \mathbf{Z}$ corresponds to a subset of formulas that can potentially influence X. Note that for each $Z \in \mathbf{Z}$, we recover the influencing formula that Z corresponds to. We now compute an average vector for Z as follows. Let $v_1 \ldots v_k$ be vectors corresponding to each active formula in Z. That is, for a non-relational formula v_i corresponds to the embedding that was learned for the feature object within that formula and for a relational formula, v_i corresponds to the embedding learned for the object that X is connected to within that formula.

For example, the formula embedding for a non-relational formula $\texttt{Word}(W, X) \Rightarrow \texttt{FakeNews}(X)$ is the embedding learned for W and for a relational formula $\texttt{FakeNews}(X)$, $\wedge$ $\texttt{SameUser}(X, X')$ $\wedge$ $\texttt{FakeNews}(X')$, it is the embedding learned for X'. We compute the average embedding corresponding to the binary vector Z as $T(Z) = \frac{1}{|\mathbf{Z}|} \sum_i^k v_i$. We now learn importance weights for the formulas using $T(Z)$. Specifically, let $g()$ be a linear function. We learn the coefficients for $g()$ using $f(T(Z))$ as a label for $g(Z)$. That is, we want to know how the presence or absence of a formula encoded within Z influences $f()$. A larger coefficient value in $g()$ denotes that the formula corresponding to that coefficient is more important to the prediction $f(X)$. Specifically, We learn the coefficients by minimizing the following loss function.

$$\mathcal{L}(f, g, \pi_x) = \sum_{Z \in \mathbf{Z}} \pi_x(T(Z))(f(T(Z) - g(Z))^2 \tag{6.1}$$

where $\pi_x(T(Z))$ quantifies the distance between the object to be explained X and the vector $T(Z)$. Specifically, $\pi_x(T(Z))$ is an exponential kernel defined on the distance $v_X^\top T(Z)$,where v_X is the embedding for X. The learned coefficients for $g()$ quantify the importance of formulas in the prediction $f(X)$.

SHAP

SHAP [48] is yet another popular outcome explainer. The main difference here is that a SHAP explanation quantifies importance based on *Shapley* values. Specifically, let $\mathbf{S}$ denote all possible subsets of formulas that contain X which is the query object whose prediction needs to be explained. Given a classifier $f()$, we compute $f(X)$ with a specific formula included and excluded from $\mathbf{S}$ and measure its difference. The exact Shapley value which quantifies the contribution of X' is computed by aggregating this difference over all $S \in \mathbf{S}$. Clearly, this is infeasible in practice since the number of possible subsets of is exponential in the number of formulas containing X. Therefore, we use the approach called KernelSHAP [48] to efficiently approximate Shapley values.

Similar to LIME, we approximate the classifier with a simpler model based on Eq. (6.1). Note that the explanation depends upon the choice of $g()$ and π_x. It is shown in [48] that the

Shapley values can be obtained through weighted linear regression similar to LIME, where we learn the coefficients for $g()$, assuming that g is a linear model. However, in LIME, the weight $\pi_x(T(Z))$ that indicates the distance between the object to be explained X and the sampled binary vector (Z) that denotes the formulas that influence X is computed based on the similarity between them. That is, we use the vector distance between v_X and $T(Z)$. In KernelSHAP, we use a special kernel to determine the weights for solving the weighted linear regression problem from which the coefficients of $g()$ are determined. Specifically, this is given by the following equation.

$$\pi_x(C(X)) = \frac{|\bar{x}| - 1}{\binom{|\bar{x}|}{|C(X)|}(|\bar{x}| - |C(X)|)} \tag{6.2}$$

where $\bar{x}$ denotes all the formulas that X participates in and $C(X) \subseteq \bar{x}$ is called a *coalition* of formulas, i.e., a subset of formulas containing X. The equation indicates that large or small coalitions a have larger weights. This is because a small coalition highlights the individual influence of each formula and a large coalition highlights the influence of a formula in collaboration with other formulas. Both are equally important in the explanation for $f(X)$. Thus, we sample coalitions of formulas and weight them according to the kernel. We then solve the weighted linear regression problem to determine coefficients that quantify the influence of the formulas in $\bar{x}$.

6.3 Explanation Feedback

Since explanations are designed for humans, it is hard to arrive at an "optimal" explanation without a human's perspective. This is particularly important since explainers such as LIME/SHAP are approximate methods and the type of explanations depend upon the quality of the approximation. Specifically, in LIME, the explanations depend upon the sampled binary vectors and in SHAP, they depend upon the coalitions that are used in the explanation. Thus, using human knowledge within these explainers can help generate more meaningful explanations.

To do this, based on a user's opinion, we generate positive/negative labels for explanations on a small subset of instances. Then, similar to the approach in unsupervised label propa-

gation [102], we propagate these labels to other instances. Specifically, we cluster the instances into K clusters and from each cluster, we sample instances for feedback to cover a diverse set of instances. Let $\{\mathbf{C}_i\}_{i=1}^{K}$ be K clusters learned using the embeddings for the instances. Thus, all objects within the same cluster have similar embeddings which means these objects are approximately symmetric to each other. We sample instances from each cluster, explain the sampled instances and ask a user to label the explanation. Let b_i be the label for the explanation on instance i, where $b_i = 1$ if the user liked the explanation or 0 otherwise. We now propagate the labels to all instances based on the label propagation approach. Specifically, a label is propagated to its neighbors in the embedding until we reach consensus. Thus, each neighborhood in the embedding is labeled with similar labels (1/0). This indicates that for similar instances, based on the feedback by the user, the explanations are likely to be either interpretable or not.

We next compute importance weights for MLN formulas based on the propagated feedback labels. Specifically, let f be a relational formula that connects X_i and X_j (analogously for non-relational formulas, f is a formula that connects X_i with a feature object). We compute the importance weight for f by predicting if X_j is likely to be a good explanation for X_i. Specifically, let $X_i^* = \arg\max_{X^* \in \mathbf{X}^*} v_{X_i}^\top v_{X^*}$, where $\mathbf{X}^*$ is the set of instances for which the user has provided feedback. Thus, X_i^* represents the most similar instance (closest neighbor) to X_i for which we have human feedback. Inspired by an approach that is commonly used in analogical reasoning which is a well-known application in word embeddings [51], we predict explanation vectors for X_i based on the explanations given to X_i^*. Specifically, let the explanation to X_i^* consist of X_{i1}^* $\ldots$ X_{iM}^*. Let $\alpha_1 \ldots \alpha_M$ be real-values signifying the relative importance of each instance in the explanation. The predicted vector that best explains X_i based on X_{ik}^* is $p_{ik} = v_{X_i^*} - v_{X_{ik}^*} + v_{X_i}$. Thus, p_{ik} is the predicted explanation vector for X_i based on an explanation to X_i^*.We compute the similarity $v_{X_j}^\top p_{ik}$ which denotes how close a potential explainer X_j is to the predicted explanation vector. To take into account all predicted explanations for X_i^*, we find the max-weighted similarity over all the instances that explain X_i^*. That is, $s_{ij} = \max_k \alpha_k v_{X_j}^\top p_{ik}$. Thus, s_{ij} is an importance weight for the formula connecting X_j and X_i that quantifies how likely is X_j to be a good (user-

Algorithm 4: Explanations in Relational Data

Input: Data $\mathbf{X}$, MLN $\mathcal{M}$
Output: Explanations
// Learn Embeddings
$O =$ Contexts for $\mathbf{X}$ from active formulas in $\mathcal{M}$
$E =$ Learn embeddings from O using Word2Vec models
// Explain Instances
$\mathbf{C} =$ Cluster $\mathbf{X}$ into K clusters using E as features
for *each cluster* C **do**
    Sample X from C
    $L(X) =$ LIME/SHAP explanation for X using E $b =$ Human feedback label for $L(X)$
// Propagate Feedback
Propagate the labels across $\mathbf{X}$
for *each* X *in* $\mathbf{X}$ **do**
    Activate formulas containing X based on Eq. (6.3)
$O' =$ Contexts for $\mathbf{X}$ from active formulas
$E' =$ Learn embeddings from O' using Word2Vec models
// Explanations
for *each* X *in* $\mathbf{X}$ **do**
    $L(X)$ = LIME/SHAP explanations for X using E'

interpretable) explanation for X_i based on the current feedback. Given the importance weights for all formulas where X_i occurs, we activate the formulas based on the following probability.

$$P(X_i, X_j) = b_i * s_{ij} + (1 - b_i) * (1 - s_{ij}) \tag{6.3}$$

From the above equation, if X_i has a positive label from the label propagation, we activate the relational formula connecting X_i and X_j with probability s_{ij} and if X_i has a negative label, we activate it with a probability $(1 - s_{ij})$. That is, for positive labels, we focus attention on formulas with larger importance scores while for negative labels, we tend to focus attention on formulas with smaller importance weights. For the MLN with modified activations, we then regenerate the embeddings and generate the final explanations using LIME and SHAP based on the new embeddings. Algorithm 1 summarizes the main steps in our approach.

6.4 Experiments

Through our experiments, we try to answer the following questions. i) Does feedback help improve the overall quality of explanations? and ii) Does relational knowledge help improve the overall interpretability of explanations? To answer these questions, we perform experiments for three text classification problems with annotated data as well as a user study.

6.4.1 Datasets

We used three datasets in our evaluation. All of them correspond to text classification. The first dataset is a dataset sampled from Yelp [64] containing 1544 reviews. The task here is to classify if a review is a positive ($\geq$ 4 stars) or negative one ($\leq$ 2 stars). We refer to this dataset as reviews dataset. Our second dataset consists of 650 COVID tweets from Kaggle. We classify whether a COVID-19 tweet contains useful information or not. We manually annotated tweets based on whether they contain facts that can be considered as useful information (e.g. scientific details, policies, etc.) or if the tweets are non-informative. We refer to this dataset as the Covid dataset. Our third dataset uses a portion of the well-known 20newsgroups dataset from the UCI repository. Specifically, we had 1000 articles and the task is to classify articles as those related to automative topics or other topics. We refer to this as the topics dataset. Note that all our tasks are binary classification tasks and we had a balanced dataset for all our tasks, i.e., roughly, there were an equal number of instances corresponding to each of the 2 classes for each dataset.

6.4.2 Relational Knowledge

For each dataset, we have a corresponding MLN that encodes relational knowledge in the dataset. The MLN encodes two types of formulas. Feature formulas that connect words to a query. Specifically, this encodes the well known bag-of-words model, where for an object X (that corresponds to a specific instance), we have formulas of the form $\texttt{Word}_w(X) \Rightarrow \texttt{Positive}(X)$, which is a relationship between a specific word w and the query X. That is, presence of w in X asserts that

X belongs to the positive class (since our tasks are binary classification tasks). we can use object X and query atom $\texttt{Positive}(X)$ interchangeably here since $\texttt{Positive}(X)$ is a singleton atom). Thus, the feature formulas for X contain all the non-relational features used to classify X. Further, we have relational formulas that connect different query objects/instances. Specifically, we encode the homophily relationship between pairs of queries. The general form for this relationship is $X \wedge \texttt{Linked}(X, Y) \Rightarrow Y$. That is, if two instances X and Y are linked together then they are likely to belong to the same class. The exact definition of $\texttt{Linked}(X, Y)$ depends on the dataset. For the reviews data, two reviews X and Y are linked if they are written by the same user or are written about the same restaurant. Similarly for the COVID data, two tweets are linked if they are from the same user and for the topics data, two articles are linked if they are from a common user.

6.4.3 Implementation

We implemented our approach using Gensim [65] to learn the embeddings [29]. We generated explanations using the embeddings in LIME and SHAP. An explanation for an instance X is a ranking over words in the text as well as other instances related X. For feedback, we used K-Means clustering to cluster the embeddings and for each cluster, we explained a positive and negative instance closest to the center. We then obtained feedback from 5 graduate students for these explanations and then used a simple majority vote toe decide the label for an explanation (good/bad explanation). We then used this to re-sample features in the instances and then generated the final explanations for all instances.

Further, to compare extrinsic explainers such as LIME/SHAP with explanations generated as part of the model itself, we generated explanations using attention in deep neural networks [91]. Though attention is viewed as being distinct from explanations [30], recent work has shown that in attention can provide meaningful explanations [98]. Therefore, we used BERT [14] which is arguably the most popular attention models for text data. Note that in this case, the explanations are limited to just words and no relational explanations are generated. We added feedback to BERT using the same approach we used for the other methods. Specifically, once we active a subset of

non-relational formulas based on feedback, we masked words that are not a part of these active formulas in our input to BERT.

In the rest of this section, we will use the following mnemonics to refer to different approaches. `RLIME` is the LIME model augmented with relational embeddings, `RFBLIME` is the LIME model with relational embeddings and using feedback from users. Analogously, for the SHAP explainer, we have `RSHAP` and `RFBSHAP` explanation methods. We refer to attention-based explanation method as `BERT` and the one where we use feedback as `RFBBERT`.

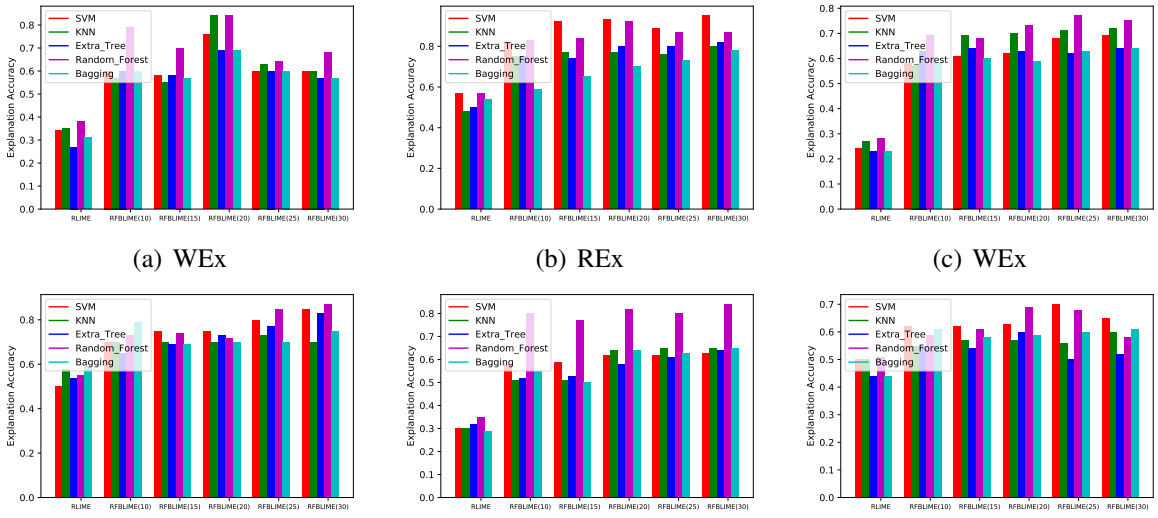
6.4.4 Annotated Explanations

To obtain ground truth for explanations, we annotated the 3 datasets. Specifically, we annotated each instance with the 5 most important words that are representative of that instance's class. The annotation was performed by graduate students who were well-versed with XAI. To remove biases, we asked 3 graduate students to independently annotate the the data. We then asked each of them to look at the other 2 rankings and through a small group discussion arrive at a consensus on the final explanations for each instance. To annotate relational explanations, we used Gensim's Doc2Vec to find 5 nearest neighbors to instances. Specifically, for each text instance, we vectorized the instance through Doc2Vec. We then determined the explanation for an instance X as the 5 nearest neighbors to X as determined by Doc2Vec.

6.4.5 Accuracy of Explanations

We computed the accuracy of explanations using the annotated explanations as ground truth. Specifically, to explain an instance X, the explainers generate a list of most important words in X and a ranking over other instances linked to X. We consider the top 5 words and the top 5 related instances as the overall explanation. We measured the $\%$ of matches in the generated explanations given the annotated explanation of X to measure the accuracy in explaining X. We average the accuracy over all test instances to obtain the accuracy score for the full dataset.

We evaluated the effectiveness of several classifiers within LIME and SHAP. Fig. 6.3 shows

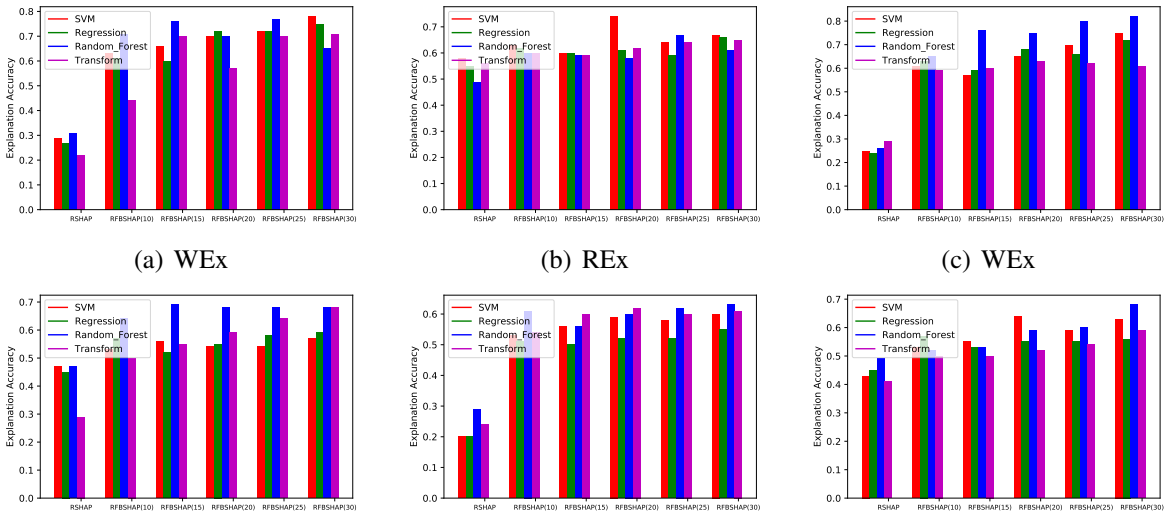


(a) WEx (b) REx (c) WEx

(d) REx (e) WEx (f) REx

Figure 6.3: Accuracy scores for different datasets using the LIME explainer. WEx refers to word explanations and REx to relational explanations. (a), (b) are results for the reviews data, (c), (d) for Covid and (e), (f) for Topics. For RFBLIME, the results are shown for varying number of clusters (specified in brackets).

the accuracy of explanations using 5 classifiers in LIME for all 3 datasets. The classifiers used were Support Vector Machines (SVMs), Random Forests (RF), Extra Trees (ET), K-Nearest Neighbors (KNNs) and Bagging (BG). As seen here `RFBLIME` significantly outperforms `RLIME` illustrating that user feedback helps us generate more interpretable explanations. This is consistent as we increase the number of clusters. Further, the RF classifier gave us the best explanations for both `RLIME` and `RFBLIME` followed by SVMs. The accuracy of `RFBLIME` for the reviews dataset was higher since sentiment words are typically retained (and non-sentiment words are removed) based on the feedback resulting in better explanations. For the Covid data, since tweets have a smaller number of words it is possible that we may remove important words as a result of the feedback resulting in slightly decreased accuracy compared to reviews. The results shown in Fig. 6.4 are similar to the LIME results and `RFBSHAP` was significantly better at expanations as compared to `RSHAP`. We evaluated with 4 classifiers in SHAP including the linear model (Regression), Random Forests (RF), SVMs and transformers. RF was generally the best classifier for explanations in `RSHAP` and `RFBSHAP`.



(a) WEx (b) REx (c) WEx

(d) REx (e) WEx (f) REx

Figure 6.4: Accuracy scores for different datasets using the SHAP explainer. WEx refers to word explanations and REx to relational explanations. (a), (b) are results for the reviews data, (c), (d) for Covid and (e), (f) for Topics. For RFBSHAP, the results are shown for varying number of clusters (specified in brackets).

6.4.6 Information in Explanations

Here, we evaluate if the explanations produce information that is useful for classification. Specifically, we measure the accuracy of classification as we incrementally add features in the explanation. That is, if the explanation is generating informative features, then the accuracy of classification is reasonably good even using only the features that constitute the explanation. This is similar to the Explanation Information Curves (EIC) metric used in [34]. Table 6.1 shows the results where we add 10% of the total features incrementally in ranked order of importance as given by the explainer. We then used the best performing classifier (Random Forests) to learn a classifier using only these features. The table shows the peak average 5-fold cross validation F1-score achieved by this classifier as we add the features incrementally. We also show the mean and standard deviation of the F1-score. As a baseline, the average 5-fold cross validation accuracy of Random Forests using all the features for the three datasets are as follows. Reviews (0.79), Covid (0.84) and Topics (0.86). As shown by the results in Table 6.1, in all cases, `RFBLIME` and `RFBSHAP` perform better than `RLIME` and `RSHAP` indicating that the explanations produced by `RFBLIME` and `RFBSHAP`

Dataset	Method	Top-Exp-Acc (F1)	Peak-Acc (F1)	Std-Dev
Reviews	RSHAP	0.45	0.52	0.03
	RLIME	0.51	0.54	0.03
	RFBSHAP	0.62	0.69	0.03
	RFBLIME	0.61	0.63	0.02
Covid	RSHAP	0.45	0.57	0.03
	RLIME	0.41	0.54	0.04
	RFBSHAP	0.55	0.61	0.01
	RFBLIME	0.57	0.60	0.03
Topics	RSHAP	0.43	0.54	0.02
	RLIME	0.55	0.58	0.07
	RFBSHAP	0.72	0.77	0.01
	RFBLIME	0.70	0.72	0.04

Table 6.1: Average F1-scores for classification as we add only the explanations incrementally in ranked order. The Peak score shows the best score obtained, and the average and standard deviation are also shown for the scores.

are of more value to the classifier. We obtained the best results for the topics dataset, where the explanation produced the best classification results.

6.4.7 Comparing Explanations and Attention

Table. 6.2shows the comparison between attention and explanations.Specifically, it only uses the accuracy of word explanations since the attention-based methods do not generate relational explanations. As shown in the table, the word explanations for BERT is lesser than RLIME and RSHAP for reviews but much better for the other 2 datasets. However, the overall accuracy for all three methods is low. RFBLIME, RFBSHAP and RFBBERT perform much better since the feedback helps us focus on important words in the explanation. RFBBERT works best in the Covid dataset compared to the other datasets.

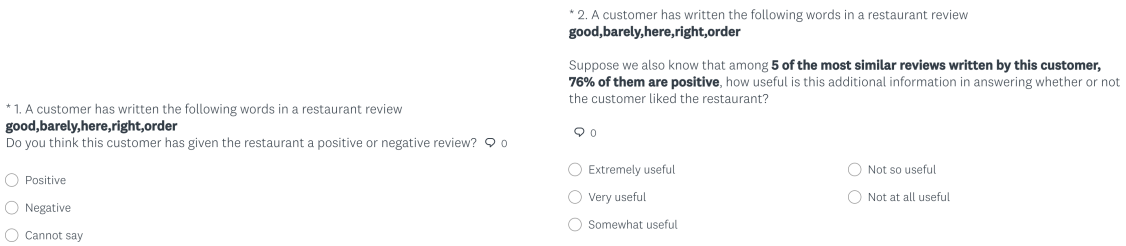
Dataset	BERT	RLIME	RSHAP	RFBBERT	RFBLIME	RFBSHAP
Reviews	0.32	0.39	0.33	0.74	0.84	0.77
Covid	0.44	0.29	0.27	0.90	0.77	0.80
Topics	0.52	0.35	0.29	0.70	0.82	0.67

Table 6.2: Comparing the accuracy of attention and explanation methods.

6.4.8 User Study

We evaluate the efficacy of explanations through a user study. We recruited 100 paid participants spread across all ages in our user study (participants paid through surveymonkey). We designed a survey to measure if using relational dependencies and feedback improves the usability of explanations. Specifically, we used the review dataset in the study and first asked the user to try to predict the class for the review (positive or negative review) using only the top-5 words ranked by the explanation. The user also had the option of choosing "cannot say" if they were unsure of whether the review was positive or negative. Next, for the same review, we presented to the user the relational explanation, and asked them to rate if this was helpful in choosing the class for the review. The rating was on 5-point Likert scale. To ensure that a non-expert user understood the relational explanation, we translated it to English as follows. If you are given additional information that in N other reviews written by this customer, he/she has rated $K\%$ as positive. Here N is the size of the relational explanation, i.e., for any review X, we choose the top-8 other reviews as ranked by the explainer in its relational explanation. To ensure tat we take into account the importance weights assigned by the explainer, we computed $K = \frac{W_+}{W_+ + W_-}$, where W_+ is the sum of weights assigned by the explainer to the positive reviews among the N reviews and W_- is the sum of weights assigned by the explainer to the negative reviews among the N reviews. We randomly selected 4 positive and 4 negative reviews for the survey. We sent the `RLIME` and `RFBLIME` to one group of users and the `RSHAP` and `RFBSHAP` to a second group. An example of the survey questions is shown in Fig. 6.5. Here, we used the best configuration for the explainers using Random Forests to generate the explanations.

The results from the user study are shown in Fig. 6.6. As seen from Fig. 6.6 (a) and (b)



(a) (b)

Figure 6.5: Survey questions with (a) assessing if a user can obtain the class using just the word explanations and (b) assessing if relational explanations are important to a user.

show the percentage of participants who chose each value in the Likert scale indicating the usefulness of the relational explanation. As shown by the results, for RFBLIME and RFBSHAP, the relational explanations had greater impact since the importance weights for the relational explanations change when we re-sample based on the feedback. Thus, with feedback, for an instance X, its relational explanation has other instances such that instances with the same class as X have larger importance weights. In the comparison between RFBLIME and RFBSHAP, participants found RFBSHAP to be either extremely useful or very useful. Next, we evaluate how likely a user is to provide the right class based on the explanation words and the results are shown in Fig 6.6 (c) and (d). As shown here, users are more likely to predict the class correctly when the explanations are from RFBLIME and RFBSHAP. This is true both for predicting the positive class from the explanation as well as predicting the negative class. The $\%$ of people who got the negative class instances correct was slightly smaller than those for the positive class. Finally, Fig. 6.6 (e) and (f) show the percentage of participants who were uncertain about the class(for both negative and positive classes). The results show that the uncertainty is reduced when we use feedback since the word explanations become more precise and it is easier for users to be certain of which class the review belongs to based on the word explanations.

To test for statistical significance, we performed the two sided t-test for our survey results. Specifically, we first tested how significant feedback is for the relational explanations. To do this, we used the rating scores given by users on the Likert scale for RLIME and RFBLIME. The average rating for RLIME was 3.04 and the average rating for RFBLIME was 3.46 and the p-value

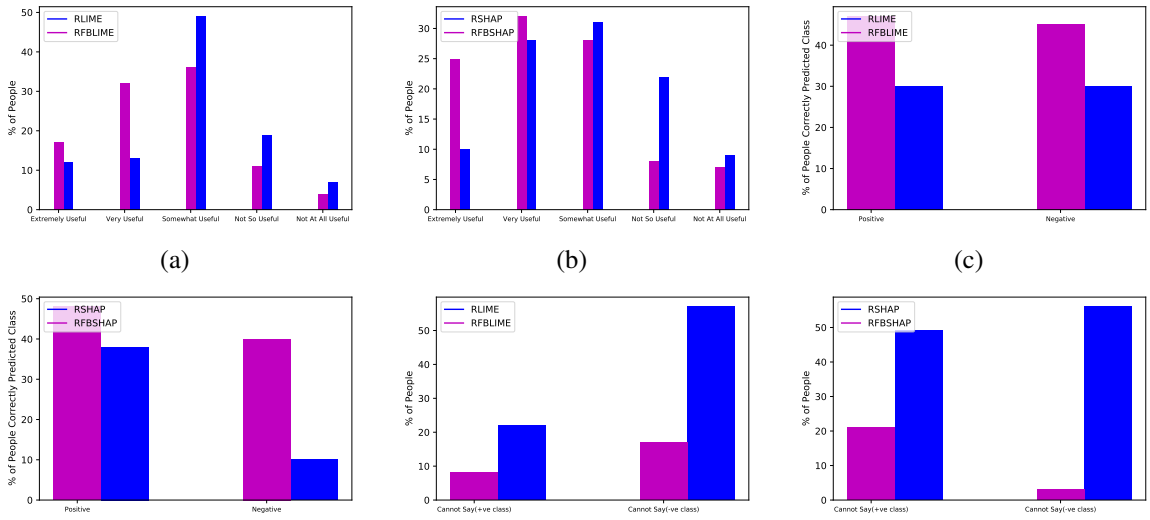


Figure 6.6: Results from the user study.

was 0.001 showing that users did prefer RFBLIME to RLIME and the difference was statistically significant. For RSHAP and RFBSHAP, the mean ratings were 3.06 and 3.63 with the p-value in the t-test lesser than $10^{-}5$, which again shows that users preferred RFBSHAP and the difference in ratings was statistically significant. Further, we also verified how well users could predict the correct class using the explanations. For this, we coded the correct answer, i.e., the case where the user could correctly guess the positive or negative class based on the explanation as 1, and the wrong answer or if the option cannot-say was selected as 0. Thus a larger number of 1's indicate that the explanations were more helpful for the user in making the right prediction of the class. For RLIME, the average for these scores was 0.17 while for RFBLIME, it was 0.72 and the p-value for the t-test was less than 10^{-20} indicating that the explanations by RFBLIME was significantly better than RLIME in terms of helping users make the correct prediction of the class. For RSHAP and RFBSHAP, the results were similar with the averages being 0.3 and 0.76 respectively with the p-value being less than 10^{-19}, thus clearly illustrating the improvement made by RFBSHAP. In summary, with our user study, we concluded that i) users prefer to have relational information as part of the explanation and ii) feedback helps improve the usability of explanations.

6.5 Summary

Explaining the results of predictions made by an AI model is extremely important to build trust in the model. A popular approach to do this is to try to approximate a hard-to-explain black-box classifier with a simpler model which can be interpreted more easily. Well known explanation methods such as LIME and SHAP use this approach. However, when the data has relationships among its instances, we can exploit these relationships to generate richer explanations. In this chapter, we modeled relationships using a symbolic AI language namely Markov Logic Networks (MLNs), and used these relationships to generate explanations. Specifically, we learn embeddings from the formulas in the MLN and use these embeddings within explainers such as LIME and SHAP. Further, since explanations are subjective, there is an inherent human element in explanations. Therefore, we used feedback from a sample of the explanations and then propagated this feedback to other symmetrical instances. We showed through a comprehensive evaluation for three text classification tasks that using both relationships and feedback significantly improves the quality of explanations. Further, through a user study, we showed that richer explanations that uses relationships significantly enhances the perception of explanations among users.

Chapter 7

Future Work

The development of Explainable AI (XAI) is absolutely critical for the success of future AI-based applications. In this dissertation, we explored XAI with the help of a symbolic AI model, namely Markov Logic Networks (MLNs). In particular, we focused on explaining relational data where the relationships are defined using the MLN. There are a number of possible future directions in this space. We outline a few possibilities below.

7.0.1 Model Explanations for Relational Data

In this dissertation, we mainly focused on explaining individual outcomes. That is, given a query, we explain why a particular prediction was made for that query. Such outcome explanations give the user a sense of trust and understanding of the prediction as is seen by our results. Further, it is possible to explain individual predictions even if the model is highly complex. Specifically, in so-called *black-box* models, we can still try to explain individual predictions. However, in this case, an explanation of the full model is still missing. That is, a user cannot understand how the model behaves in general. For example, in a decision tree classifier, the model itself is easily explainable regardless of the instance we are trying to classify. In general, for many complex models, it is difficult if not infeasible to understand how the model would behave for different types of inputs. In particular, such explanations can be causal explanations [59]. Here, we can identify specific variables in the model that cause certain types of predictions. On the other hand, we could also have contrastive explanations [43] to better explain the model. That is, humans in general prefer explanations where we specify an alternate hypothesis. Specifically, why a prediction was made

instead of another prediction. For example, contrasting a healthy with a not healthy patient in terms of specific attributes such as weight, blood-sugar level, etc. helps a humans establish a frame of reference. Further, another type of related explanation is counterfactual explanations [43] where we tend to think of how the output prediction would have been different if the input was modified. For example, how many pixels should I have changed in this image to change the prediction of the classifier.

We can extend our work for model explanations in relational data. Some possible directions include, finding relationships that are contrastive or counterfactual in nature. That is, we can try to explain a model as follows. We replace a subset of relationships with an alternate relationships in the data and quantify the effect on the overall distribution. A major challenge here is to scale up to large relational models. For instance an MLN can contain thousands of ground formulas where each formula encodes a logical relationship. Finding counterfactual explanations in such a large space may be challenging and we approximations may be required to do this.

7.0.2 Fairness in AI

A related aspect of AI that has received significant attention is fairness in AI [50]. Here, the idea is identify if the model is fair from bias towards specific groups of individuals. For instance, a commonly used example is that image classifiers are trained with more male subjects and therefore they have better accuracy on images containing male subjects as compared to female subjects. The techniques proposed in this dissertation can play a role towards achieving fairness in AI systems. Specifically, one possible direction is to model fairness constraints as formulas in the MLN. We can then verify if these constraints explain the inferences made by the model. Note that we could potentially encode complex fairness criteria compared to simple features in this case. For example, in non-relational classifiers, we could try to point to specific features and check for presence or absence of the features in an explanation as a validation that the classifier is being fair or not. However, a single feature may not be sufficient to encode complex criteria. Instead, we could use first-order logic to specify the criteria for fairness. Of course, simply using the fairness constraints

could still lead to predictions that are not fair. However, it would still improve trust in the model since the explanation would point out to the importance of the fairness constraints in decision making.

7.0.3 Interactive Explanations

A third possible direction to extend this dissertation is to develop interfaces for explanations. Specifically, we showed that using human feedback can significantly improve explanations. Therefore, we can build an interface where a user can specify his/her preferences for explanations and we can generate a unique model that suits these preferences. The user can then manually adjust the model such that it generates explanations that the user can easily interpret. Thus, in general we can dynamically construct the model starting from the type of explanations needed for a particular user (or task).

Chapter 8

Conclusion

Explainable Artificial Intelligence (XAI) has rapidly grown to establish itself as one of the key research areas in AI. In this dissertation, we focused on outcome explanations for relational data. Specifically, outcome explanations explain predictions made by the model. We used the language of Markov Logic to build XAI models. Specifically, Markov Logic Networks (MLNs) are a symbolic AI model that encodes soft logical relationships among variables. However, even though MLNs are symbolic, explaining their predictions is a hard problem since it involves performing inference over very large models. Therefore, we developed techniques that can efficiently use MLNs for XAI. Specifically, we first developed an approach that can simplify a large MLN using *tied-parameter* learning, where we combined functions in the MLN distribution. Next, we developed a new approach to explain the marginal probabilities in MLNs based on probabilities generated from Gibbs sampling. The approach ranks the formulas in the MLN in order of importance in the marginal probability inference task. However, it turns out that in large MLNs, this is problematic since Gibbs samples are not as reliable. Specifically, the mixing time of the sampler for large MLNs is infeasibly large and therefore, the explanations generated may not be as interpretable. Therefore, we developed a novel approach where we use symmetries to simplify an MLN into a number of smaller MLNs where the sampler is more reliable. We then generate explanations from these simpler MLNs and unify them into a global explanation. Finally, we explored the utility of human feedback in relational explanations. Specifically, since explanations are inherently subjective, it is often hard for explainers to generate the explanations with the right information. Therefore, we developed a novel approach where we use the relationships encoded in the MLN to identify sym-

metrical instances and then obtain human feedback for a small subset of explanations. We then propagate the feedback based on the symmetry between instances. We applied our general idea to generate relational explanations from several well-known model-agnostic explanation methods such as LIME, SHAP and also a model-specific explanation using the attention mechanism in deep networks. In all our approaches, we conducted comprehensive evaluations including quantitative metrics as well as user studies. We showed that relationships in the data play an important role in enhancing the quality of explanations. While there are a number of open challenges in XAI, we hope that this dissertation is a step towards making progress in this all important topic.

References

[1] Babak Ahmadi et al. “Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training.” In: *Machine Learning* 92.1 (2013), pp. 91–132.

[2] Ankit Anand et al. “Contextual Symmetries in Probabilistic Graphical Models.” In: *International Joint Conference in Artificial intelligence*. 2016, pp. 3560–3568.

[3] Guy van den Broeck and Adnan Darwiche. “On the Complexity and Approximation of Binary Evidence in Lifted Inference.” In: *Advances in Neural Information Processing Systems 26*. 2013, pp. 2868–2876.

[4] Guy Van den Broeck et al. “On the Tractability of SHAP Explanations.” In: *AAAI*. 2021.

[5] Hung Hai Bui, Tuyen N. Huynh, and Sebastian Riedel. “Automorphism Groups of Graphical Models and Lifted Variational Inference.” In: *UAI*. 2013.

[6] Nadia Burkart and Marco F. Huber. “A Survey on the Explainability of Supervised Machine Learning.” In: *J. Artif. Intell. Res.* 70 (2021), pp. 245–317.

[7] Li Chou et al. “On Parameter Tying by Quantization.” In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. Phoenix, Arizona: AAAI Press, 2016, pp. 3241–3247.

[8] Mark Craven and SeÃ¡n Slattery. “Relational Learning with Statistical Predicate Invention: Better Models for Hypertext.” In: *Machine Learning* 43.1/2 (2001), pp. 97–119. URL: http://dblp.uni-trier.de/db/journals/ml/ml43.html#CravenS01.

[9] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[10] Adnan Darwiche and Auguste Hirth. “On The Reasons Behind Decisions.” In: *CoRR* abs/2002.09284 (2020).

[11] L. De Raedt, A. Kimmig, and H. Toivonen. “ProbLog: A Probabilistic Prolog and Its Application in Link Discovery.” In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. 2007, pp. 2462–2467.

[12] R. de Salvo Braz. “Lifted First-Order Probabilistic Inference.” PhD thesis. Urbana-Champaign, IL: University of Illinois, 2007.

[13] R. Dechter. “Bucket elimination: A unifying framework for reasoning.” In: *Artificial Intelligence* 113 (1999), pp. 41–85.

[14] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In: *NAACL-HLT (1)*. Association for Computational Linguistics, 2019, pp. 4171–4186.

[15] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. San Rafael, CA: Morgan & Claypool, 2009.

[16] Khan Mohammad Al Farabi et al. “Fine-Grained Explanations Using Markov Logic.” In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD*. 2019.

[17] Mohammad Khan Al Farabi, Somdeb Sarkhel, and Deepak Venugopal. “Efficient Weight Learning in High-Dimensional Untied MLNs.” In: *AISTATS*. 2018, pp. 1637–1645.

[18] Ruth C. Fong and Andrea Vedaldi. “Interpretable Explanations of Black Boxes by Meaningful Perturbation.” In: *ICCV*. IEEE Computer Society, 2017, pp. 3449–3457.

[19] S. Geman and D. Geman. “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6 (1984), pp. 721–741.

[20] Michael R. Genesereth and Eric Kao. *Introduction to Logic, Second Edition*. Morgan & Claypool Publishers, 2013.

[21] L Getoor and B Tasker. *Introduction to statistical relational learning MIT Press*. 2007.

[22] Bhavya Ghai et al. "Explainable active learning (xal): An empirical study of how local explanations impact annotator experience." In: *arXiv preprint arXiv:2001.09219* (2020).

[23] V. Gogate and P. Domingos. "Probabilistic Theorem Proving." In: *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2011, pp. 256–265.

[24] V. Gogate, A. Jha, and D. Venugopal. "Advances in Lifted Importance Sampling." In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.

[25] Riccardo Guidotti et al. "A Survey of Methods for Explaining Black Box Models." In: *ACM Comput. Surv.* 51.5 (2018), 93:1–93:42.

[26] David Gunning. "DARPA's explainable artificial intelligence (XAI) program." In: *IUI*. 2019.

[27] Jan Van Haaren et al. "Lifted generative learning of Markov logic networks." In: *Machine Learning* 103.1 (2016), pp. 27–55.

[28] Geoffrey E. Hinton. "Training Products of Experts by Minimizing Contrastive Divergence." In: *Neural Computation* 14.8 (2002), pp. 1771–1800.

[29] Mohammad Maminur Islam, Somdeb Sarkhel, and Deepak Venugopal. "On Lifted Inference Using Neural Embeddings." In: *AAAI*. 2019, pp. 7916–7923.

[30] Sarthak Jain and Byron C. Wallace. "Attention is not Explanation." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 3543–3556.

[31] Stefanie Jegelka, Suvrit Sra, and Arindam Banerjee. "Approximation Algorithms for Tensor Clustering." In: *Algorithmic Learning Theory, 20th International Conference, ALT*. 2009, pp. 368–383.

[32] Shaoxiong Ji et al. "A Survey on Knowledge Graphs: Representation, Acquisition and Applications." In: *CoRR* abs/2002.00388 (2020).

[33] Nitin Jindal and Bing Liu. "Opinion Spam and Analysis." In: *Proceedings of the 2008 International Conference on Web Search and Data Mining*. 2008, pp. 219–230.

[34] A. Kapishnikov et al. "XRAI: Better Attributions Through Regions." In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 4947–4956.

[35] H. Kautz, B. Selman, and Y. Jiang. "A General Stochastic Approach to Solving Problems with Hard and Soft Constraints." In: *The Satisfiability Problem: Theory and Applications*. Ed. by D. Gu, J. Du, and P. Pardalos. New York, NY: American Mathematical Society, 1997, pp. 573–586.

[36] Tushar Khot et al. "Exploring Markov Logic Networks for Question Answering." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. 2015, pp. 685–694.

[37] Pang Wei Koh and Percy Liang. "Understanding Black-box Predictions via Influence Functions." In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2017, pp. 1885–1894. URL: http://proceedings.mlr.press/v70/koh17a.html.

[38] S. Kok et al. *The Alchemy System for Statistical Relational AI*. Tech. rep. http://alchemy.cs.washington.edu. Seattle, WA: Department of Computer Science and Engineering, University of Washington, 2006.

[39] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[40] Todd Kulesza et al. “Principles of explanatory debugging to personalize interactive machine learning.” In: *Proceedings of the 20th international conference on intelligent user interfaces*. 2015, pp. 126–137.

[41] S. L. Lauritzen and D. J. Spiegelhalter. “Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems.” In: *Journal of the Royal Statistical Society. Series B (Methodological)* 50.2 (1988), pp. 157–224.

[42] Shili Lin and Jie Ding. “Integration of ranked lists via cross entropy Monte Carlo with applications to mRNA and microRNA Studies.” In: *Biometrics* 65 (2009), pp. 9–18.

[43] Zachary Chase Lipton. “The Mythos of Model Interpretability.” In: *CoRR* abs/1606.03490 (2016).

[44] Rachel Lomasky et al. “Active class selection.” In: *European Conference on Machine Learning*. Springer. 2007, pp. 640–647.

[45] D. Lowd and P. Domingos. “Efficient Weight Learning for Markov Logic Networks.” In: *Principles of Knowledge Discovery in Databases*. Warsaw, Poland, 2007, pp. 200–211.

[46] D. Lowd and P. Domingos. “Recursive random fields.” In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. Hyderabad, India: AAAI Press, 2007, pp. 950–955.

[47] Scott M Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions.” In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[48] Scott M. Lundberg et al. “From local explanations to global understanding with explainable AI for trees.” In: *Nature Machine Intelligence* 2.1 (2020), pp. 56–67.

[49] A. McCallum, K. Nigam, and L. Ungar. “Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching.” In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2000, pp. 169–178.

[50] Ninareh Mehrabi et al. “A Survey on Bias and Fairness in Machine Learning.” In: *ACM Comput. Surv.* 54.6 (2021), 115:1–115:35.

[51] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality.” In: *NIPS*. 2013, pp. 3111–3119.

[52] Tim Miller. “Explanation in artificial intelligence: Insights from the social sciences.” In: *Artif. Intell.* 267 (2019), pp. 1–38.

[53] Happy Mittal et al. “Fine Grained Weight Learning in Markov Logic Networks.” In: *Workshop on Statstical Relational AI*. 2016.

[54] Brent Mittelstadt, Chris Russell, and Sandra Wachter. “Explaining Explanations in AI.” In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 2019, 279â€“288.

[55] M. Mladenov, A. Globerson, and K. Kersting. “Efficient Lifting of MAP LP Relaxations Using k-Locality.” In: *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics* (2014).

[56] Mathias Niepert. “Markov Chains on Orbits of Permutation Groups.” In: *UAI*. AUAI Press, 2012, pp. 624–633.

[57] Mathias Niepert and Guy Van den Broeck. “Tractability through exchangeability: A new perspective on efficient probabilistic inference.” In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence, AAAI Conference on Artificial Intelligence*. 2014.

[58] Feng Niu et al. “Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS.” In: *PVLDB* 4.6 (2011), pp. 373–384.

[59] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge, UK: Cambridge University Press, 2000.

[60] D. Poole. “First-Order Probabilistic Inference.” In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. Acapulco, Mexico: Morgan Kaufmann, 2003, pp. 985–991.

[61] H. Poon and P. Domingos. “Joint Inference in Information Extraction.” In: *Proceedings of the 22nd National Conference on Artificial Intelligence*. Vancouver, Canada: AAAI Press, 2007, pp. 913–918.

[62] H. Poon and P. Domingos. “Joint Unsupervised Coreference Resolution with Markov Logic.” In: *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, HI: ACL, 2008, pp. 649–658.

[63] H. Poon and P. Domingos. “Sound and Efficient Inference with Probabilistic and Deterministic Dependencies.” In: *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. This volume. Boston, MA: AAAI Press, 2006.

[64] Shebuti Rayana and Leman Akoglu. *Yelp Dataset for Anomalous Reviews*. Tech. rep. http://odds.cs.stonybr Stony Brook University, 2015.

[65] Radim Řehůřek and Petr Sojka. “Software Framework for Topic Modelling with Large Corpora.” In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. 2010, pp. 45–50.

[66] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Anchors: High-Precision Model-Agnostic Explanations.” In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2018.

[67] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “"Why Should I Trust You?": Explaining the Predictions of Any Classifier.” In: *Knowledge Discovery and Data Mining (KDD)*. 2016.

[68] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “"Why Should I Trust You?": Explaining the Predictions of Any Classifier.” In: *Knowledge Discovery and Data Mining (KDD)*. 2016.

[69] Matthew Richardson and Pedro Domingos. “Markov logic networks.” In: *Machine learning* 62.1-2 (2006), pp. 107–136.

[70] Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. “Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations.” In: *IJCAI*. ijcai.org, 2017, pp. 2662–2670.

[71] D. Roth. “On the Hardness of Approximate Reasoning.” In: *Artificial Intelligence* 82 (1996), pp. 273–302.

[72] Chiradeep Roy et al. “Explainable Activity Recognition in Videos.” In: *Joint Proceedings of the ACM IUI Workshops*. 2019.

[73] Stuart J. Russell and Peter Norvig. “Artificial Intelligence : A Modern Approach.” In: Malaysia; Pearson Education Limited, 2016.

[74] Christopher De Sa et al. “Rapidly Mixing Gibbs Sampling for a Class of Factor Graphs Using Hierarchy Width.” In: *Advances in Neural Information Processing Systems 28*. 2015, pp. 3097–3105.

[75] Somdeb Sarkhel et al. “Efficient Inference for Untied MLNs.” In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. 2017, pp. 4617–4624.

[76] Somdeb Sarkhel et al. “Lifted MAP inference for Markov Logic Networks.” In: *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS-14)* (2014).

[77] Somdeb Sarkhel et al. “Scalable Training of Markov Logic Networks Using Approximate Counting.” In: AAAI, 2016.

[78] R. R. Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization.” In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626.

[79] Burr Settles. “Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances.” In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. 2011, pp. 1467–1478.

[80] Xiaoting Shao et al. “Right for Better Reasons: Training Differentiable Models by Constraining their Influence Functions.” In: *AAAI*. 2021, pp. 9533–9540.

[81] Vishal Sharma et al. “Lifted Marginal MAP Inference.” In: *UAI*. AUAI Press, 2018, pp. 917–926.

[82] Andy Shih, Arthur Choi, and Adnan Darwiche. “A Symbolic Approach to Explaining Bayesian Network Classifiers.” In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*. 2018, pp. 5103–5111.

[83] P. Singla and P. Domingos. “Discriminative Training of Markov Logic Networks.” In: *Proceedings of the Twentieth National Conference on Artificial Intelligence*. Pittsburgh, PA: AAAI Press, 2005, pp. 868–873.

[84] P. Singla and P. Domingos. “Entity Resolution with Markov Logic.” In: *Proceedings of the Sixth IEEE International Conference on Data Mining*. Hong Kong: IEEE Computer Society Press, 2006, pp. 572–582.

[85] P. Singla and P. Domingos. “Lifted First-Order Belief Propagation.” In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*. Chicago, IL: AAAI Press, 2008, pp. 1094–1099.

[86] Alison Smith-Renner et al. “No explainability without accountability: An empirical study of explanations and feedback in interactive ml.” In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–13.

[87] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks.” In: *ICML*. 2017, pp. 3319–3328.

[88] Stefano Teso and Kristian Kersting. “"Why Should I Trust Interactive Learners?" Explaining Interactive Queries of Classifiers to Users.” In: *CoRR* abs/1805.08578 (2018).

[89] Son D. Tran and Larry S. Davis. “Event modeling and recognition using markov logic networks.” In: *ECCV*. 2008.

[90] G. Van den Broeck et al. “Lifted Probabilistic Inference by First-Order Knowledge Compilation.” In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. 2011, pp. 2178–2185.

[91] Ashish Vaswani et al. “Attention is All you Need.” In: *Neural Information Processing Systems*. 2017, pp. 5998–6008.

[92] D. Venugopal and V. Gogate. “On Lifting the Gibbs Sampling Algorithm.” In: *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*. 2012, pp. 1664–1672.

[93] Deepak Venugopal and Vibhav Gogate. “Evidence-based Clustering for Scalable Inference in Markov Logic.” In: *ECML PKDD*. 2014.

[94] Deepak Venugopal, Somdeb Sarkhel, and Kyle Cherry. “Non-parametric Domain Approximation for Scalable Gibbs Sampling in MLNs.” In: *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI 2016, June 25-29, 2016, New York City, NY, USA*. 2016.

[95] Deepak Venugopal, Somdeb Sarkhel, and Vibhav Gogate. “Just Count the Satisfied Groundings: Scalable Local-Search and Sampling Based Inference in MLNs.” In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[96] Deepak Venugopal et al. “Relieving the Computational Bottleneck: Joint Inference for Event Extraction with High-Dimensional Features.” In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL, 2014, pp. 831–843.

[97] Lodewyk F. A. Wessels and Etienne Barnard. “Avoiding false local minima by proper initialization of connections.” In: *IEEE Trans. Neural Networks* 3.6 (1992), pp. 899–905.

[98] Sarah Wiegreffe and Yuval Pinter. “Attention is not not Explanation.” In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 11–20.

[99] J. S. Yedidia, W. T. Freeman, and Y. Weiss. “Generalized Belief Propagation.” In: *Advances in Neural Information Processing Systems 13*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Cambridge, MA: MIT Press, 2001, pp. 689–695.

[100] Quanshi Zhang and Song-Chun Zhu. “Visual Interpretability for Deep Learning: a Survey.” In: *Arxiv* (2018). URL: http://arxiv.org/abs/1802.00614.

[101] Zhiqiang Zheng and Balaji Padmanabhan. “On active learning for data acquisition.” In: *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* IEEE. 2002, pp. 562–569.

[102] Xiaojin Zhu and Zoubin Ghahramani. *Learning from Labeled and Unlabeled Data with Label Propagation*. Tech. rep. 2002.