UNIVERSITY OF CALIFORNIA
SANTA CRUZ

INFERRING LOCAL AND GLOBAL PROPERTIES IN KNOWLEDGE GRAPHS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

by

Varun R Embar

December 2021

The Dissertation of Varun R Embar
is approved:

Lise Getoor, Chair

Jeffrey Flanigan

Christos Faloutsos

Peter F. Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Inferring local and global properties in Knowledge Graphs

by

Varun R Embar

Understanding the meaning, semantics and nuances of entities and the relationships between entities is crucial for enabling future generations of AI systems to go beyond keyword-based pattern matching. Knowledge bases provide AI systems with a structured representation of entities, their attributes and their relationships. Knowledge graphs (KGs) are a type of knowledge base that uses a graph to store information. They have become ubiquitous as they provide efficient storage and retrieval. A wide range of systems such as search engines, intelligent agents, contextual recommender systems and fake news detection applications use KGs as a knowledge source. To perform effectively, these systems need to extract latent patterns in the KG that are novel, valid and useful, a task known as *knowledge discovery*. In my thesis, I examine three key tasks of knowledge discovery - *data alignment*, which involves inferring relationships between entities of the same type, computing *aggregate graph queries*, which involves estimating recurring subgraphs in the absence of information and *model discovery*, a data-driven way of discovering and combining rules to reason in a KG. These tasks are challenging because: (1) KGs contain a rich set of entities and relations that have very different characteristics and also highly correlated (2) KGs are large, containing millions of entities, and at the same time incomplete, missing many entities and relationships (3) lack of training data and hard to generate negative instances for training various approaches.

In this dissertation, I develop robust and scalable knowledge discovery algorithms to address these challenges. First, I develop a data alignment approach that identifies both entities that are exactly the same, and also *variations*. Variations are entity that are similar in most aspects but differ in a few aspects. The proposed approach is a scalable and unsupervised. Using empirical evaluation in three different domains I show the generality of this approach. Second, I propose a fine-grained data alignment approach to identify discriminating attributes between variations. The proposed approach can identify a rich variety of discriminating attributes for different entity types. The framework models the semantics of the attributes enabling it to scale to a large number of attributes with little training data. Third, I develop a scalable framework to estimate global graph properties using complex graph queries. The proposed approach estimates these queries when there is missing information, such as node labels. I analyse two different approaches, point estimate and expectation-based approaches, for this task both empirically and theoretically. Finally, I present a framework to perform model discovery in knowledge graphs. The framework can also generate explanations for the model's prediction. To discovery these models efficiently, I first propose a template-based rule mining technique that can efficiently search the space of rules. I then propose a new scoring function that enables the framework to learn the relative importance of these rules efficiently. Finally, I prove the stability of that the generated explanations. Together my work expands the scope of knowledge discovery tasks on knowledge graphs.

Acknowledgments

First and foremost, I would like to thank my advisor, Lise Getoor, who has been a constant pillar of support, and without whom this PhD would not have been possible. Her counsel and encouragement have been my constant companions even before the start of my PhD. I still fondly remember the conversation that I had with her when I was deciding between universities for my PhD. If I have been able to navigate the hard times during my PhD, it is only because I was standing on her giant shoulders. Her constant feedback and comments, attention to details and the push to give my best, have not only made me a better researcher, but also greatly improved me as a person. It is only after interacting with Lise have I come to truly appreciate the Sanskrit hymn “acharya devo bhava”.

Special thanks to my advancement committee members Snigdha Chaturvedi, Luna Dong and Jim Whitehead. Your feedback and guidance were the foundation blocks on which this dissertation was built. I would also like to thank Jeffrey Flanigan and Christos Faloutsos for accepting to be part of my defense committee. Christos, in particular, has been a constant mentor throughout my PhD.

Next, I would like to thank my collaborators both in the LINQS lab and at Amazon, without whom my PhD would not have shaped the way it has. The summer internships at Seattle provided me the first real breakthrough during my PhD. I am fortunate to have worked under the able guidance of Luna Dong, Bunyamin Sisman, Hao Wei, Christos Faloutsos and Andrey Kan. They were incredibly patient and allowed me explore different avenues during my internships. I would like to express my gratitude to Sriram Srinivasan who has instrumental in helping me break the gridlocks in my research. My work on aggregate graph queries and structure learning would not have seen the light of day if it weren’t for him. The late night conversations and coding sessions were instrumental in meeting the deadlines. I would also like to thank Jay Pujara, Dhanya Sridhar and Golnoosh Farnadi for their mentorship during the early days of my PhD.

I was able to navigate this new country only due to the help of LINQS lab members. I would like to thank Eriq Augustine, Charles Dickens, Connor Pryor, Vibin Vijay, Shresta B.S, Vihang Godbole, Sabina Tomkins, Pigi Kouki, Rishika Singh, Caleb Levy, Niharika Srivastav, Alex Miller, Kyle Fredrickson, Yatong Chen, Jason Ting, Dhawal Jhorapurkar, Nikhil Kini, Johnnie Chang, Hung Ju and Shobeir Fakhraei for your constant companionship. I knew I could always reach to Eriq when I couldn't get my code to work as intended. I have learned a lot from you and hope to code like you someday. I would be ignorant of many board games if it weren't for Connor's board game nights. My proofs wouldn't be correct if Charles hadn't gone over them. Special thanks to Cynthia Mccarley without whom I would have drowned in paper work.

I would like to thank my roommates Sriram Srinivasan and Parameshwaran Raman for all the late night technical and non-technical conversations, and for putting up with my cooking. I would also like to thank Rohini, Prashanth, Vishnu, Achyuth, Meera, Vasu, Omshree, Rajat, Pragnya, Kathik and Preeti for being my family outside home. I knew I could always bank on you for a good home-cooked meal. I would like to thank my parents Ravikumar and Koustubha, my sister Varsha and my brother-in-law Sandeep for believing me and keeping me sane. Finally, I would like to thank my Masters advisor, Indrajit Bhattacharya, and my manager at IBM, Vinayaka Pandit, without whom I would not have embarked on my PhD journey.

Part of this work was supported by the National Science Foundation grants CCF-1740850, CCF-2023495 and IIS-1703331, AFRL, IFDS DMS-2023495, the Defense Advanced Research Projects Agency and an Amazon Research Award.

Chapter 1

Introduction

Gathering, cleaning, organizing and storing large amounts of useful data in structured knowledge bases is a fundamental challenge of AI. Knowledge bases enables sophisticated AI applications to query data in realtime for performing reasoning and planning tasks. Initial AI approaches organised the data using closed ontologies, vocabularies, and agreed upon schemas[11]. These approaches mainly concentrated on structured sources of data present in relational tables. A challenge with these systems was that the schemas needed to evolve constantly and this breaks down when extracting information at a large scale.

The rapid growth of the World Wide Web has brought with it an explosion in the amount of data available online. A wealth of knowledge is hidden in the semi-structured (e.g Wikipedia infoboxes) and unstructured (e.g. blogs, user comments and reviews) data found on the web. Recent advances in machine learning and information extraction have allowed us to tap into these sources leading to the *crossing of structure chasm* [98]. These large knowledge bases store the extracted information in the form of graphs called *knowledge graphs* [26, 55, 166, 149]. Knowledge graphs represent entities as nodes in the graphs and relationships between these entities are stored as typed edges.

Knowledge graphs help systems understand and reason about various real-world entities, and have become an integral part of a variety of systems including search engines, intelligent agents and fake news detection applications. Knowledge graphs have enabled search engines to return structured information along with search results in the form of knowledge panels [84]. Intelligent agents such as Alexa or Siri return the exact answer the user is looking for with the help of knowledge graphs [137]. Knowledge graphs enable factoid question answering systems, such as IBM Watson, to defeat humans in quiz shows such as Jeopardy![104]. Fake news detection algorithm use knowledge graphs to verify the veracity of news content [182].

In order to perform these tasks, these systems need to extract novel, valid and useful patterns from the knowledge graph, a task known as *knowledge discovery*. While significant progress has been made in this regard, several challenges still exist. In this dissertation, I explore the existing approaches, enumerate challenges faced in the process, and develop methods and provide insights to tackle such challenges.

1.1 Opportunities

Knowledge graphs provide a structured representation of entities, their attributes and their relationships using a graph [191]. Most knowledge graphs follow the RDF standard[106] and represent facts about entities using triples of the form (*Subject*, *Predicate*, *Object*). While subject and object are entities, the predicate denotes the type of relationship that exists between the two entities. For example ($WashingtonD.C.$, $capitalOf$, *United States*) is a triple which encodes the fact that Washington D.C. is the capital of United States. The entities in the knowledge graph are usually typed. The relationships that entity types participate in and other related meta-information is defined by an ontology. Recent advances in information extraction have led to the construction of many large scale knowledge graphs [26, 55, 166, 149]. I give more detailed background on knowledge graphs in the next chapter.

These knowledge graphs provide a structured source of knowledge which is important for a wide range of systems. Knowledge graphs allow AI systems to go beyond matching keyword on semi-structured and unstructured text. They enable these systems to understand the semantics, nuances and meaning of entities present in text.[224]. They also provide relationships that exist between different these entities. While some knowledge graphs deal with entities such as people and places [26, 166], others such as Cyc[142] and ConceptNet [149] focus on commonsense knowledge. Several domains specific knowledge graphs such as Bio2RDF[17] and LinkedLifeData[167] for biology and Product Graph[56] for e-commerce applications have also been constructed.

Several state-of-the-art recommendation systems use knowledge graphs to not only to generate better recommendations and also to provide users with an explanation for each recommendation [261, 38]. Question answering systems and chatbots reason over knowledge graphs or use knowledge graph embeddings to accomplish their task [210, 110]. Fake news detection techniques use knowledge graph embeddings or run graph mining algorithms on a knowledge graphs to identify trustworthy sources and to check the veracity of the facts [182, 44, 58].

To accomplish these tasks, these systems discover local patterns and relationships the exist between individual entities in the graph and also global properties that often involve several or all entities and relationships in the knowledge graphs. The local patterns could be relationships that hold between two entities of the same type or those that hold across entity types. One important relationship that can be discovered between entities of the same type is the *equivalence* relationship that identifies entities that are the same. For example, a product knowledge graph may contain entities corresponding to products present across various retailers' catalogs. This is related to the problem of entity resolution in knowledge graphs [61, 192]. Apart from this, we can also discover other relationships between entities of the same type such *variations*, where the entities are similar to each other in some aspects but differ in other aspects. We can further identify the aspects along which the entities differ in. I refer to the problem of discovering these rich set of relationships between entities of the same type in knowledge graphs as *data alignment*.

Estimating global properties such as the importance of an entity and identifying the presence of recurring subgraph patterns in the knowledge graph is useful for tasks such as schema generation[144], identifying strong ties [208] and for learning representations of the graph [262]. Discovering recurring subgraphs is related to the task of *network motif* mining [165]. The presence of these subgraph patterns can be computed using complex graph queries. However, since knowledge graphs are incomplete, we need to infer the missing information before computing them. I refer to the task of estimating these subgraphs using complex graph queries as *aggregate graph queries*.

As nodes in a knowledge graphs represent realworld entities, these nodes also follow global patterns that are inherent to realworld entities. For example, we know that for all entities A, B and C, if the relationships $motherOf(A, B)$, $brotherOf(C, A)$ is true, then the relationship $uncleOf(C, B)$ must be true. These patterns can be represented using logical rules such as:

$$motherOf(A, B) \wedge brotherOf(C, A) \rightarrow uncleOf(C, B)$$

Discovering these latent rules enables construction of models that can reason in a knowledge graphs. Discovering such rules from the facts present in a knowledge graphs is known as *rule mining* [81, 79, 82, 49]. For reasoning in knowledge graphs, several such rules need to be mined, evaluated and combined before arriving at a decision. I refer to this task of mining and combining several related rules to construct a model that can reason in a knowledge graphs as *model discovery*. This is similar to the task of inductive logic programming, where logical theories are discovered from instances and background knowledge [169]. This problem is also related to the task of *structure learning* in statistical relational learning [89]. In the next sections, I talk about the various challenges related to tasks of data alignment, estimating aggregate graph queries and model discovery, and provide insights and methods to tackle these challenges.

1.2 Challenges

There are several challenges that need to be address when aligning entities, estimating graph queries and discovering models in knowledge graphs.

Rich set of entities and relations: Knowledge graphs model realworld and contain a rich set of entity and relation types often running into the hundreds. Different entity types have very different characteristics, participate in different relations and can even have different set of output labels for the same task. For example, variant alignment relations such as *size* and *color* can be inferred for products such as clothing, where as *flavor* and *package size* can be inferred for products such as groceries. Models that reason in a knowledge graphs must combine and evaluating a large set of entities and relations and need to be highly scalable.

Missing or incomplete information: Knowledge graphs are large, containing millions of entities and relationships, and at the same time incomplete and noisy, with many relationships about these entities missing. Knowledge discovery approaches need to be scalable and at the same time robust to missing information and noise. For example, when identifying patterns or rules in the data, we may mine rules that are biased. Consider a rule mining algorithm run on a knowledge graphs with contains facts about people, most of whom live in United States. The algorithm might mine the rule $isPerson(A) \rightarrow livesIn(A, UnitedStates)$. Knowledge discovery approaches need to be robust to handle such biases.

Lack of negative instances: Knowledge graphs only encode information about true facts and do not contain facts that are false. Most learning approaches require both positive and negative facts for training. Some of the approaches overcome this challenge by making a *local closed-world* assumption, and assumes all missing facts about a given entity and a predicate is false. However, since knowledge graphs are known to be incomplete, this assumptions might not hold in practice.

Similar and correlated relations: Relations that entities participate in are highly correlated and different entity types participate in similar but different relations. Knowledge discovery approaches need to infer these correlations. For example, *length* and *breadth* are highly similar relations that are correlated. *Color* and *finish* are similar relations but different entity types participate in these relations.

Little or no training data: Due the presence of large number of entity types and relations, often there is little or no training data to train models. Unsupervised approaches or approaches that require fewer training instances need to be developed for these tasks.

1.3 Contributions

My current thesis addresses the above mentioned challenges for the tasks of data alignment, estimating aggregate graph queries and model discovery. My work on data alignment is motivated by product knowledge graph used in the e-commerce domain. I use citation graphs to evaluate various proposed approaches for the task of estimating aggregate graph queries. I show the advantages of the proposed explainable model discovery framework using recommendation graphs that deal with users and items. Below I highlight the key contributions of my work. Portions of this dissertation have been published elsewhere [71, 72, 73, 69].

In Chapter 3, I look at the task of data alignment for nearly identical, but distinct, entities called *entity variations*. Entity variations share the same value for key attributes such as brand, manufacturer and product line, but differ in other attributes, which I call *variational attributes*, such as package size and color. Identifying these variational attributes is crucial for aligning entities with equivalence and variation relationships. However, these variational attributes are domain dependent and often present only in unstructured text. To address this challenge, I introduce the notion on *contrast features* and propose a novel unsupervised approach, **VarSpot**, to mine them from unstructured text. **VarSpot** reasons about both similarities and differences between entities, and can easily scale to large sources containing millions of entities. I show the generality of my approach by performing experimental evaluation on three different domains. I then use these mined contrast features to perform **contrastive entity linkage**, an approach that aligns entity pairs with variation and equivalence or duplicate relationships. The proposed approach significantly outperforms state-of-the-art learning-based and rule-based entity linkage systems when identifying duplicates and variations. Further, through annotations using Mechanical Turk, I show the interpretable nature of contrast features. This work was published at the Automated Knowledge Base Construction (AKBC) 2020.

Next, in Chapter 4, I develop a fine-grained alignment framework that identifies the discriminating attributes between entity variations. Identifying these discriminative attributes between entity variations, e.g., the same wristwatch models but in different finishes, is crucial for improving e-commerce search engines and recommender systems. Despite the importance of these discriminative attributes, as in the case of most variational attributes, they are often not available explicitly and instead are mentioned only in unstructured fields. To address this challenge, I introduce the novel task of *discriminative attribute extraction* which involves identifying the attributes that distinguish product variations, such as finish, and also, extracting the values for these attributes from unstructured text. This task differs from the standard attribute value extraction task that has been well-studied in literature, as the discriminating attribute needs to be identified in addition to finding the values for this attribute. I propose a novel end-to-end, deep learning based approach, **DiffXtract**, that jointly identifies both the discriminative attribute and extracts its values from the product variations. The proposed approach is trained using a multitask objective and explicitly models the semantic representation of the discriminative attribute and uses it to extract the attribute values. This allows the approach to identify wide range of discriminative attribute across domains with very little training data. I show that the existing attribute extraction approaches have several drawbacks, both theoretically and empirically. I also introduce a novel dataset based on a corpus of data previously crawled from a large number of e-commerce websites. Through empirical evaluation, I show that **DiffXtract** outperforms state-of-the-art deep learning-based and dictionary-based attribute extraction approaches when identifying both the attribute and when extracting attribute values. This work was published at the International Conference on Big Knowledge (ICBK) 2021.

In Chapter 5, I propose the concept of aggregate graph queries that capture a class of global graph properties such as cluster cohesion and the number of bridge nodes. These properties are crucial to glean insights about a graph's community structure and spread of influence. Efficiently computing graph properties when the graph is fully observed has received significant attention [250, 45], however the problem of computing aggregate graph queries when there is missing information has received little attention. Computing these queries for graphs with missing attributes involves performing inference over the graphs. I study the effectiveness of statistical relational learning and graph neural networks in estimating these queries. I estimate these properties first using point estimates. For approaches that can infer a joint distribution over the missing attributes, I also estimate these properties as an expectation over the distribution. I theoretically show that for even simple queries on graphs with two nodes, point estimates cannot minimize the expected mean square error. To compute the expectation tractably for probabilistic soft logic, one of the statistical relational learning approaches, I introduce a novel sampling framework. In the experimental evaluation, using three benchmark datasets, I show that expectation-based approaches tend to outperform point estimates when computing aggregate graph queries. This work was published in International Joint Conference on Learning & Reasoning (IJCLR) 2021.

In Chapter 6, I develop a framework to perform model discovery in knowledge graphs. I propose a new structure learning algorithm, Explainable Structured Model Search (*ESMS*), that learns a model consisting of weighted logical rules and also provides a framework to generate explanations for the predictions generated by the model. The proposed approach discovers models that trade-off predictive accuracy and explainability. Learning these rules from data comes with high computational costs as it typically requires an expensive combinatorial search over the space of rules and repeated optimization over rule weights. *ESMS* uses a novel search procedure to efficiently search the space of models. The piecewise pseudolikelihood (PPLL) objective used by the procedure does not require re-optimizing the rule weights across models during each iteration of the search. Further, I introduce the notion of *relational stability* and prove that the proposed explanation framework is stable. In the empirical evaluation on three realworld datasets, I show that the proposed approach not only discovers models that are explainable, but also significantly outperforms existing state-out-the-art structure learning approaches. This is under review and a part of this work was published at Statistical Relational AI (StarAI) 2018.

1.4 Organization

This dissertation is organised as follows. In Chapter 2, I formally introduce knowledge graphs and provide the necessary background to understand statistical relational learning and graph neural network approaches. The next two chapters deal with local properties of the graph that hold between a pair of entities. In Chapter 3, I propose an approach to align entities with duplicate and variation relations. In Chapter 4, I refine the variation relationship by identifying the discriminative variational attribute. Chapter 5 and Chapter 6 deal with global properties of the graph. In Chapter 5, I propose an approach to estimate aggregate graph properties when some entity labels are missing. In Chapter 6, I propose an explainable model discovery framework. Finally, in Chapter 7, I summarize the contributions of my thesis and provide future directions to expand this work.

Chapter 2

Background

Three areas of research are important to my work. First, I give an overview of knowledge graphs. Next, I discuss the area of statistical relational learning(SRL) that combines probabilistic reasoning with knowledge representation. I talk about Markov Logic Networks(MLN) and Probabilistic Soft Logic(PSL), two statistical relational learning frameworks that I use in this dissertation. Finally, I describe deep learning based graph neural networks for inference in relational data.

2.1 Knowledge Graphs

Knowledge Graphs(KG) provide a structured representation of entities, relationship and attributes using a graph. Most knowledge graphs loosely follow the RDF standard and represent facts about entities using triples of the form (*Subject*, *Predicate*, *Object*). While subject and object are entities, the predicate denote the type of relationship that exists between the two entities.

While the initial KGs were hand curated and small[142, 164], advances in information extraction have fueled the growth of many large scale knowledge graphs such as NELL[166], Yago[230], Freebase[26] and Knowledge Vault[55] to name a few. Some knowledge graphs focus on encoding facts about proper entities such as people and places [26, 166], others such as Cyc[142] and ConceptNet [149] focus on commonsense knowledge. Several domains specific knowledge graphs such as Bio2RDF[17] and LinkedLifeData[167] for biology and Product Graph[56] for e-commerce applications have also been constructed.

Knowledge graphs encode facts about entities in the form of triples. While the presence of a triple suggests a true fact, the interpretation of the absence of a triple differs based on the KG paradigm. In *closed-world* paradigm, missing triples are interpreted as facts that are false. However, in *open-world* paradigm, missing triples are interpreted as facts where the truth value is unknown.

Entity resolution, link prediction and link-based clustering of related entities are some of the tasks that are well studied with respect to KGs in literature [179]. *Entity Resolution* refers to the problem of matching entity mentions in structured and unstructured data. For a survey of entity resolution approaches refer to Brizan and Tansel [30] and Getoor and Machanavajjhala [88]. In the context of KGs, entity resolution refers to matching pairs of entities across KGs that refer to the same real-world entity. Pujara and Getoor [192], Rastogi et al. [202], Singla and Domingos [225], Pujara et al. [193] are some of the approaches proposed for entity resolution in KGs. *Link prediction*, also called *knowledge graph completion* is the task of inferring missing triples in the knowledge graph. Proposed techniques for this task can be classified as graph-based algorithms [248, 85, 140] and embedding or tensor-based algorithms [178, 258, 28]. *Clustering of related entities* refers to the task of grouping entities in relational data based on their similarity. In the social network domain, this task is known as community detection[77]. Saeedi et al. [209], Gad-Elrab et al. [80] are some examples of clustering approaches for knowledge graphs.

In this thesis, I propose a richer formulation for these three tasks involving multiple entities and relations and propose novel approaches to solve them.

Data alignment: Data alignment refers to the task of discovering semantic relations such as *variations* and *discriminative attributes* between entities of the same type in a KG. The inferred relations are called *alignment relations.* Although the task of data alignment is related to the tasks of entity resolution and link prediction, it differs from both in key aspects. Entity resolution approaches infer entities that are the same. Data alignment approaches, on the other hand, infer entities that are related but are not identical. It also uses a more expressive alignment language when linking entities, denoting how the entities are related. Data alignment can be thought of as a more general form of entity resolution.

Like the task of data alignment, link prediction infer missing relations between entities. Some of the links such as *sameAs* and *subsetOf* can be thought of as aligning entities. However, link prediction is more general and can infer other relations such as *bornIn* between entities of different entity types. In contrast, data alignment always infers relations between entities of the same type.

Aggregate graph queries: Aggregate graph queries provide a rich language to identify recurring subgraphs in the graphs with incomplete information. These complex queries generally involve many nodes and edges and require joint reasoning over multiple node labels to compute them. This task combine the tasks of estimating the queries with the inference of missing information such as node labels. This is related to the task of clustering related entities. However, unlike clusters where all entities in the cluster are similar, graph queries can express more complex groups of entities using Boolean logic. Aggregate graph queries then use an aggregate function such as *average* to summarize these subgraphs.

Model discovery: Model discovery refers to the task of mining and combining logical rules, that represent complex relationships between entities and entity types, into a model that can reason in a KG. Models can reason about the existence of missing edges, type of an entity, cluster similar entities and so on. Many link prediction algorithm for KGs mine and combine rules based on observed relationships in the KG to infer new relations [139, 81]. These algorithms can considered as a special case of model discovery. Further, these rules are interpretable in nature can be used as explanations.

2.2 Statistical Relational Learning

Statistical Relational Learning (SRL) techniques combine probabilistic reasoning with knowledge representations that capture the structure in problem domains. Getoor and Taskar [89] provides a survey of various SRL models proposed in literature. Markov Logic Networks(MLN) [205] and Probabilistic Soft Logic(PSL) [14] are two popular SRL frameworks. These frameworks use weighted first-order logic rules to define an undirected graphical model and inference is performed over the generated graphical model.

Nickel et al. [179] introduce the notion of probabilistic knowledge graphs to use SRL techniques for reasoning in KGs. Given a set of entities $\mathcal{E}$ and a set of relationship $\mathcal{R}$, the probabilistic knowledge graphs represents each possible triple (e_i, r_j, e_k) by a random variable $r_j(e_i, e_k)$. These random variables $r_j(e_i, e_k)$ take values in the range $[0, 1]$ and denote the probability of the triple being true.

Before I introduce MLN and PSL formally in the next section, I give a brief overview of some of the terms used in SRL and its interpretation in the context of probabilistic knowledge graphs.

- **Predicate:** A predicate p is a relation defined by a unique identifier and a positive integer called its arity, which denotes the number of terms it accepts as arguments.

- **Atom:** An **atom** $p(\cdot)$ in SRL framework corresponds to a predicate p in the KG. At atom takes as arguments constants (e.g. `Alice, Bob`) or variables (e.g. A, B). The constants correspond to entities in the KG and the variables correspond to entity types.

- **Ground atom:** An atom whose predicate arguments are all constants is a **ground atom**. A ground atom encodes a fact or a triple, and the truth value represents the probability of the fact being true.

- Clause: A **clause** c is a formula $\wedge_i L_i \vee_j L_j$ where L_i and L_j are atoms of its negation.

- Rule: A **rule** r is a pair (w, c) where c is a clause and w is its associated real-valued weight.

- Ground rule: A rule consisting of only ground atoms is a **ground rule**.

- Model: A **model** M is a set of rules $\{R_1, R_2, \cdots, R_n\}$.

Given a set of logical rules defined by a SRL model, and a set of observations from the data, the rules are grounded to generate ground rules, where variables in a rule are substituted by all possible constants. The set of ground rules generated from rule r is denoted by G_r.

Each ground atom present in the set of ground rules is then associated with a random variable. Random variables that correspond to ground atoms with observed values are called **observed random variables(X)** and those that correspond to ground atoms with unobserved values are called **unobserved random variables(Y)**. Each ground rule is then mapped to a potential function ϕ over the random variables. The set of all potentials define a probability distribution over the random variables.

For example, consider the following rule that suggests similar items are rated similarly by the user:

$$w : SimilarItem(I_1, I_2) \wedge Rating(U, I_1) \rightarrow Rating(U, I_2)$$

Here, $SimilarItem$ is a **predicate** that encodes the similarity between the two items I_1 and I_2, and the predicate $Rating$ encodes the rating assigned to the item by the user U. w denotes the weight of the rule that determines its importance. The **variables** I_1, I_2, U range over the **constants** in a domain. Consider the set of users = {Alice, Bob} and item = {Top Gun, Valkyrie}. An example of a ground rule is as follows:

$$\begin{aligned} w :& SimilarItem(Top\ Gun, Valkyrie) \wedge Rating(Alice, Top\ Gun) \\ &\rightarrow Rating(Alice, Valkyrie) \end{aligned}$$

For the ground rule mentioned above, let X_1, Y_1, Y_2 be the random variables associated with the ground atoms $SimilarItem(Top\ Gun, Valkyrie)$, $Rating(Alice, Top\ Gun)$, $Rating(Alice, Valkyrie)$. The grounded rule is mapped to a potential $\phi(X_1, Y_1, Y_2)$.

The set of potentials defined by the ground rules is then used to define a distribution over the set of observed($\mathbf{X}$) and unobserved($\mathbf{Y}$) random variables given by:

$$P(\mathbf{X}, \mathbf{Y}) = \frac{1}{Z} \exp(\sum_{r \in M} \sum_{G_r} w_r \phi_r(\mathbf{X}, \mathbf{Y}))$$
$$\text{where} \tag{2.1}$$
$$Z = \sum_{\mathbf{X}} \exp(\sum_{r \in M} \sum_{G_r} w_r \phi_r(\mathbf{X}, \mathbf{Y}))$$

Each ϕ_r instantiated from logical rule $r \in M$ is a function that scores assignments $\mathbf{x}$. Intuitively, ϕ_r assigns higher scores to assignments that satisfy its corresponding logical rule scaled by its rule weight. Assignments that satisfy more ground rules are exponentially more probable.

2.2.1 Markov Logic Networks

Markov Logic Networks [205] are an SRL framework that define a class of undirected discrete probabilistic graphical models. An MLN model consists of weighted logical rules that encode statistical dependencies and structural constraints. It associates ground atoms with Boolean random variables. Potential functions ϕ are indicator functions that are one if a ground rule is satisfied and zero otherwise. MLN has been successfully applied to many problem including information extraction [213], entity resolution [225] and computer vision [237].

Formally, joint density function defined by an MLN is given by:

$$P(\mathbf{Y}, \mathbf{X}) = \frac{1}{Z} \exp(\sum_{r \in M} \sum_{G_r} w_r \phi_r(\mathbf{X}, \mathbf{Y}))$$
$$\text{where} \tag{2.2}$$
$$Z = \sum_{\mathbf{Y}} \exp(\sum_{r \in M} \sum_{G_r} w_r \phi_r(\mathbf{X}, \mathbf{Y}))$$

where $\phi_r(\mathbf{X}, \mathbf{Y})$ is given by:

$$\phi_r(\mathbf{X}, \mathbf{Y}) = \mathbb{1}(\mathbf{X}, \mathbf{Y}) \tag{2.3}$$

where $\mathbb{1}$ is an indicator function that is 1 if a ground rule is satisfied and 0 otherwise.

2.2.2 Probabilistic Soft Logic

Probabilistic Soft Logic(PSL) is a SRL framework and a templating language used for defining hinge-loss Markov random fields (HL-MRFs)[14], a class of undirected probabilistic graphical models, that supports modeling of multi-relational data. Like MLN, PSL model consists of weighted logical rules that encode statistical dependencies and structural constraints. However, unlike MLN where the truth values of ground atoms are Boolean, truth values in PSL are continuous valued in the range [0,1] and can be used to represent real-valued similarities. While in MLNs the rule weights are real numbers, in PSL the rule weights are positive real numbers. This make inference is PSL efficient. PSL has been successfully applied to many problem including natural language processing [18], social media analysis [115, 64] and information extraction [189].

Formally, a HL-MRF is defined by the joint density function :

$$\begin{gathered} P(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z} \exp(-\sum_{r \in M} \sum_{G_r} w_r \phi_r(\mathbf{X}, \mathbf{Y})) \\ \text{where} \\ Z = \int_{\mathbf{Y}} \exp(-\sum_{r \in M} \sum_{G_r} w_r \phi_r(\mathbf{X}, \mathbf{Y})) \end{gathered} \tag{2.4}$$

where the $\mathbf{Y}$ represent the target or unobserved ground atoms and $\mathbf{X}$ represent the observed ground atoms. G_r represents the set of all ground rules corresponding to the PSL rule r. $\phi_r(\mathbf{X}, \mathbf{Y})$ represents the linear or quadratic penalty for violating clause r based on whether $p = 1$ or $p = 2$.

2.3 Graph Neural Networks

Deep learning approaches learn non-linear hierarchical representation of the data and mapping from these representations to the final tasks [94]. These approaches alleviate the need for handcrafted features and have achieved state-of-the-art performances in several tasks ranging from object detection[203], machine translation[157], speech recognition[105] and natural language understanding[212]. Recently, there has been great interest in applying these techniques to graphs [254] in general and knowledge graphs in particular[114].

Graph neural networks can be broadly classified into recurrent graph neural networks (RecGNNs), convolutional graph neural networks (ConvGNNs), graph autoencoders (GAEs), and spatial-temporal graph neural networks (STGNNs) [256]. RecGNNs use recurrent architecture and message passing algorithms to learn node representations. ConvGNNs use the convolution operation to aggregate a node's features with the features of its neighbours. GAEs learn latent node representations that are capable of reconstructing the structural information of the graph. STGNNs learn node representation and take into consideration spatial dependency and temporal dependency.

In this dissertaion, I use three popular graph neural networks, Graph Convolutional Networks (GCNs)[122], Graph Attention Networks (GATs) [241] and Graph Markov Neural Networks (GMNNs) [197].

2.3.1 Graph Convolutional Networks

Graph Convolutional Network (GCN) [122] is a popular non-probabilistic GNN approach. GCNs iteratively update the representation of each node by combining each node's representation with its neighbors' representation. The propagation rule to update the hidden representation of a node is given by:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}H^{(l)}W^{(l)}\right) \tag{2.5}$$

where $H^{(l)}$ denotes the representation at layer l, $\tilde{D}$ represents the degree matrix, $\tilde{A}$ represents the adjacency matrix with self-loop, W represents the weights, and σ denotes an activation function, such as the ReLU. The final representations are fed into a linear softmax layer classifier for label prediction.

2.3.2 Graph Attention Networks

Graph Attention Networks (GATs) [241] are similar to GCNs and use self-attention while combining the representation of each node with its neighbors. This allows the model to assign different weights to each of its neighbors' representations. The propagation rule for GAT is given by:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}} \alpha_{ij} h_j^{(l)} W\right) \tag{2.6}$$

where $h_i^{(l)}$ is the representation of node i at layer l, W is the weight matrix, $\mathcal{N}$ is the set of neighbors and α is the attention weights.

2.3.3 Graph Markov Neural Networks

Graph Markov Neural Networks (GMNNs) [197] is a recently introduced probabilistic approach. GMNNs build on graph neural networks such as GCNs or GATs by adding a second neural network to capture the latent dependencies in the inferred data. The pair of neural networks are trained using a variational EM algorithm. In the E-step, the object representations are learned by the first neural network. In the M-step, the latent dependencies are learned by the other neural network.

Chapter 3

Data Alignment in Knowledge Graphs

3.1 Introduction

Are the two Kindle e-readers shown in Fig. 3.1 the *same* or *different*? It is unclear – while they are essentially the same product – they have the same company, brand and screen size, they also have some important differences such as color and storage. Unlike common entity linkage domains such as medical health records and publication data sets where there is a precise notion of identity, domains such as products and music are inherently more challenging due to the presence of nearly identical but distinct entities. In fact, whether these entities are distinct or not may sometimes be context dependent. We refer to these entities as *entity variations*.

More formally, entity variations are sets of entities that share the same value across core attributes called *base attributes*, but differ from each other along a few crucial attributes which we refer to as *variational attributes*. For example,

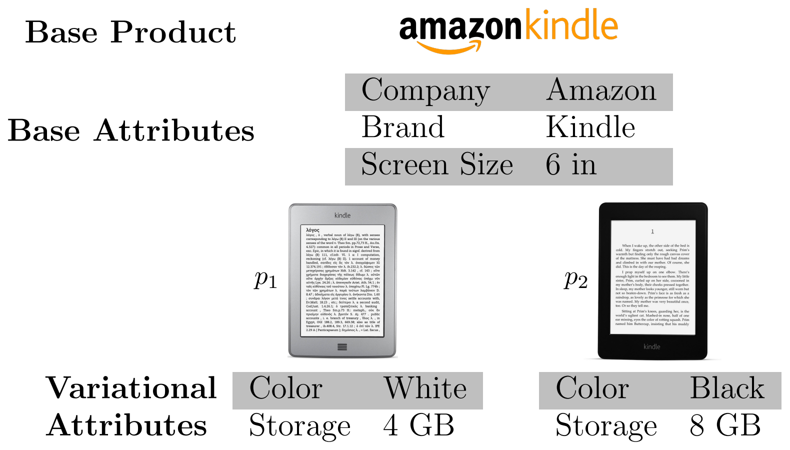


Figure 3.1: Entity variations: Variations for the tablet base entity. The base attributes include company, brand and screen size. Color, storage and option are variational attributes.

variations of the tablet in the motivating example have different values for the variational attribute storage size.

An entity linkage framework for domains with variations needs to identify both entities that are exactly the same (duplicates), and also entity variations (variations). While identifying variations is important for matching duplicate entities, it is a useful task in its own right. For example, it is critical for enhancing customer shopping experience. Consolidating product variations on the same page enables customers to locate the product they desire quickly. Grouping product variations in search results can increase diversity and prevent returned results from being dominated by multiple variants of the same product.

There has been a large body of work for the task of identifying duplicates (See [68, 129, 52, 132, 88, 43] for survey papers comparing various approaches). More recently, Li et al. [145] proposed an approach to identify entity variations in a single catalog using structured variational attributes. However, one of the main challenges in identifying variants is in the discovery of variational attributes present in unstructured text, and then jointly identifying duplicates and variations

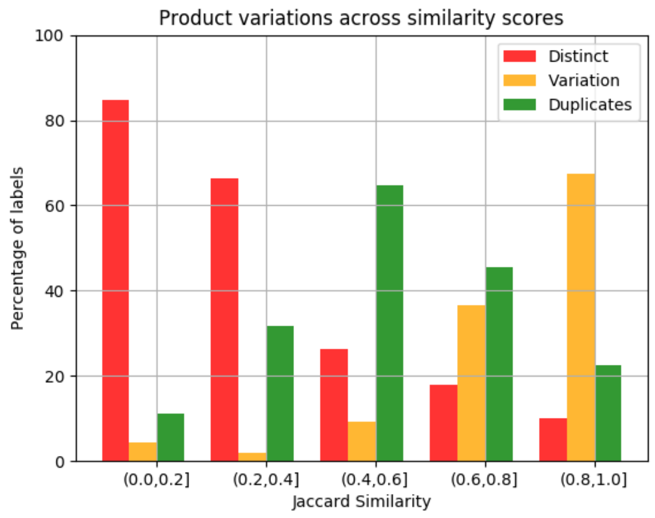


Figure 3.2: Variation histogram: The distribution of grocery product pairs which are duplicates, variations and distinct across Jaccard similarity computed on title. Even for high similarity (between $(\mathbf{0.8}, \mathbf{1.0}]$), about 70% of the pairs are actually variations.

Most data sources have few structured attributes, while many other entity attributes, including several variational attributes, are included in unstructured text such as title and description. Attribute value extraction approaches [269, 90, 187, 194, 116, 131, 206, 152] typically extract values for a small, fixed set of attributes such as brand, flavor and quantity and not all extracted attributes correspond to variational attributes. The presence of a large number of variational attributes that are different across domains limit the use of these attribute extraction techniques. In addition, these techniques are typically supervised and need labels which may be costly to acquire.

To illustrate the magnitude of the problem, consider a sampling of grocery products from two e-commerce catalogs (details described in Section 3.4). These catalogs contain two attributes, brand and title. Fig. 3.2 shows the distribution of duplicates, variations and distinct entity pairs sampled and grouped into five uniform interval buckets using Jaccard similarity computed on the title. Across

	Duplicate Matching	Variation Matching	Variational Attrib. Extraction
ER approaches	X		
Li et al. [145]		X	
Attribute extraction			X
CEL	X	X	X

Table 3.1: CEL extracts variational attributes and identifies both duplicates and variations.

all ranges, variations are mixed in with duplicates and distinct pairs, with some buckets containing up to 70% variation pairs. There are no structured attributes to help distinguish them. As we show in our experiments, even state-of-the-art approaches that compute multiple similarity measures incorrectly link variations as duplicates.

In this chapter, we address these challenges by proposing a novel framework, **contrastive entity linkage (CEL)**, that identifies both duplicates and variations together. The main contributions include:

- **Three-way linkage:** We extend the traditional task of two class entity linkage, where the goal is to identify duplicates, to a three class setting, where along with duplicates we also identify entity variations.
- **Automatic variational attribute discovery:** We proposed a variational attribute discovery approach, **VarSpot**, a scalable, unsupervised technique that analyzes similarities and differences within the *same* catalog. **VarSpot** uses the notion of *contrast features* to model variational attributes.
- **Effectiveness:** We perform empirical evaluation in three different domains to show the generality of our approach. Using three different state-of-the-art entity linkage frameworks, including rule-based and deep learning based

frameworks, we show that models with contrast features significantly outperform models without them when identifying duplicates and variations. Further, through annotations using Mechanical Turk, we show the interpretable nature of contrast features.

3.2 Preliminaries

In this section, we introduce necessary terms and notation, followed by the formal definition of three-way entity linkage.

Base entity: A *base entity* is an abstract, canonical entity associated with a set of attributes called **base attributes**.

Entity variations: *Entity variations* are a set of related entities that are instantiations of the same base entity. All entity variations share the same value for the base attributes. Entity variations are also associated with a set of **variational attributes** whose values differ across variations. We denote the function that maps an entity to its base entity by B.

Records and attributes: A *record* represents a realworld entity in a data source and is associated with a set of attributes called **record attributes**. Records usually have a combination of structured attributes such as brand and price, and unstructured text attributes such as title and description. The attribute values in a record may be missing or incorrect. Not all base and variational attributes as present as structured record attributes. We use r_i to denote a record, a_m to denote the record attribute and $r_i(a_m)$ to denote the value of the ${a_m}^{th}$ attribute of r_i. The set of all attributes is given by $\mathcal{A}$. We denote the function that maps a record to the realworld entity it represents by E.

Catalog: The set of all records present in a data source is referred to as a **catalog**, and is denoted by $\mathcal{C}$.

Definition 1. *Given two catalogs $\mathcal{C}_1, \mathcal{C}_2$, the task of* ***3-way entity linkage*** *is to identify sets of record pairs $\mathcal{M}$(Set of duplicates), $\mathcal{V}$(Set of variations) such that $\mathcal{V}, \mathcal{M} \subset \mathcal{C}_1 \times \mathcal{C}_2$, $(r_i, r_j) \in \mathcal{M} \Leftrightarrow E(r_i) = E(r_j)$, $(r_i, r_j) \in \mathcal{V} \Leftrightarrow \{B(E(r_i)) = B(E(r_j)) \wedge E(r_i) \neq E(r_j)\}$.*

Table 3.2 gives an overview of the symbols and their definitions.

Symbol	**Definition**
r_i	Record
a_m	Record attribute
$r_i(a_m)$	a_m^{th} attribute of record r_i
$\mathcal{A}$	Set of all record attributes
E	Maps record to realword entity
B	Maps entity to its base entity
$\mathcal{C}$	Catalog
$\mathcal{M}$	Set of duplicates
$\mathcal{V}$	Set of variations

Table 3.2: List of symbols used in this chapter along with their definitions.

3.3 Contrastive Entity Linkage

In this section, we introduce the notion of contrast features that capture the value of variational attributes, and propose **VarSpot**, a novel algorithm to identify them. We then describe our overall approach, **Contrastive entity linkage**, that uses **VarSpot** to identify both duplicates and variations.

3.3.1 Contrast Features

Contrast features are phrases or n-grams that represent values of variational attributes present in unstructured text attributes such as title. For the motivating example in the introduction, the phrase 4 *GB* and 8 *GB* are contrast features as they correspond to values of the variational attribute *storage size.* Since contrast

Algorithm 1 **VarSpot**: An approach to identify contrast features

```
Input: Product catalogs $\mathcal{C}_1$, $\mathcal{C}_2$
Output: Contrast features with weights $\mathbf{f}$
  # Phase 1: Link each catalog to itself to identify potential variations
  $\mathcal{L}_1 \leftarrow \textbf{Link}(\mathcal{C}_1)$
  $\mathcal{L}_2 \leftarrow \textbf{Link}(\mathcal{C}_2)$
  # Phase 2: Extract contrast features from identified variations
  $\mathbf{f_1} = \textbf{ContrastSpot}(\mathcal{L}_1)$
  $\mathbf{f_2} = \textbf{ContrastSpot}(\mathcal{L}_2)$
  # Aggregate the contrast features from the two catalogs
  $\mathbf{f} = \mathbf{f_1} \cup \mathbf{f_2}$
  return $\mathbf{f}$
```

features are attribute values of all variational attributes, they are more general. While *pack of 2, 75 oz, dark roast* are some examples contrast features in the groceries domain, *radio edit, remix, live* are examples of contrast features in the music domain.

3.3.2 Overview of VarSpot

VarSpot is a unsupervised contrast feature extraction algorithm and has two phases. In the first phase **VarSpot** identifies a set of potential entity variations in a single catalog. This phase returns a set $\mathcal{L}$ consisting of entity pairs and their similarity scores. This phase is described in Subsection 3.3.3. In the second phase, we use the **ContrastSpot** algorithm to extract phrases that distinguish pairs in $\mathcal{L}$. This is described in Subsection 3.3.4. We finally aggregate the extracted phrases from both of the catalogs to generate the contrast features. An overview of **VarSpot** is given in Algorithm 1.

3.3.3 Phase 1: Identifying Potential Variations

The first step of the proposed **VarSpot** algorithm identifies potential entity variations by linking the entities in a catalog to itself. This is motivated by two key properties of catalogs. First, catalogs typically contain very few duplicates. Most sites ensure that records in their catalog refer to distinct entities. Second, the records of variations are typically more similar to each other than to records that correspond to entities with different base products. This is due to the presence of many base attributes that are shared across variations. Based on the key properties mentioned earlier, we expect that most of the pairs with high similarity scores to be variations.

Typically, in order to limit the $O(n^2)$ potential blowup in candidate pairs, a blocking technique such as locality sensitive hashing (LSH) is used to limit the candidate pairs for linkage. For large catalogs, even after performing blocking, we may need to evaluate a large set of candidate pairs. Since entity variations share the same value for base attributes, we only need to consider record pairs that have the same values for these attributes. We block on the structured base attributes such as brand, when present in the catalog, to reduce the number of pairs.

Further, instead of using a sophisticated supervised linkage system that performs linkage using all attributes of the record, we compute a similarity measure s such as Jaccard or cosine similarity on the unstructured attribute from which we need to extract the contrast features. We then return all pairs that have score $s > \theta$.

The algorithm for identifying potential entity variations **Link** is given in Algorithm 2. The algorithm takes as input a catalog $\mathcal{C}$, structured base attributes a_b, bucket size threshold λ and similarity threshold θ. First, the algorithm buckets records based on the attribute value a_b. We then prune buckets larger than

Algorithm 2 Link: Scalable unsupervised approach to discover variations

```
Input: Catalogs $\mathcal{C}$, Base attributes $a_b$, Bucket size threshold $\lambda$, Similarity threshold $\theta$.
Output: Set of potential entity variations $\mathcal{L}$
  # Bucket entities based on base attribute $a_b$
  for $r_i \in \mathcal{C}$ do
      $H_{a_b}[r_i(a_m)] \leftarrow H_{a_b}[r_i(a_m)] \cup r_i$
  # Prune buckets with size $> \lambda$
  for $key \in H_{a_b}$ do
      if $size(H_{a_b}[key]) > \lambda$ then
          Prune $H_{a_b}[key]$
  # Compute similarity and generate pairs
  for each pair $(r_i, r_j) \in H_{a_b}$ do
      if $sim(r_i(a_v), r_i(a_v)) > \theta$ then
          $\mathcal{L} = \mathcal{L} \cup (s, r_i, r_j)$
  return $\mathcal{L}$
```

the threshold λ. For each of the remaining buckets, we consider all record pairs, compute similarity scores and output all pairs greater than threshold θ.

3.3.4 Phase 2: Extracting Contrast Features

The second phase of **VarSpot** extracts contrast features from the unstructured attributes of identified potential entity variations. The main idea is to first start with unigrams that are present in one entity but not the other, and grow the phrase by considering adjacent tokens. We continue until we find the largest *significant phrase*, and extract the phrase as a contrast feature. Similar techniques have been proposed to extract phrases in topic modeling [66].

Significant phrases are n-grams which occur more frequently than expected had they been sampled independently from their constituent sub-phrases. For example the phrase *pack of 2* could arise either because it is a phrase corresponding to a single attribute, or because *{pack of, 2}* are two independent phrases adjacent to each other. To capture this, we define a *significance score* (*sig*) and consider all phrases c with $sig(c) > \alpha$ to be a significant phrase.

The probability of observing an n-gram c can be estimated by

$$p(c) = \frac{f(c)}{L} \tag{3.1}$$

where $f(c)$ is the observed frequency of c and L is the total number of all n-grams.

A *partition* of an n-gram is a set of smaller n-grams that together form the string. For example, the partitions of the n-gram *pack of 2* are *{pack, of 2}*,*{pack of, 2}* and *{pack, of, 2}*. Let c be an n-gram, and $\mathbf{c}'$ be a partition of the tokens in the n-gram. The expected frequency of observing n-gram c due to independently sampling its constituent n-grams in **c'** is given by:

$$\hat{f}(\mathbf{c'}) = L * \prod_{t \in \mathbf{c}'} p(t) \tag{3.2}$$

where $p(t)$ is the probability of occurrence of the n-gram t. For a n-gram c and one of its partition c', the standard deviation between the observed frequency of c and the expected frequency due to independently sampling its constituent n-grams in c' is given by:

$$std\_dev(c, \mathbf{c}') = \frac{f(c) - \hat{f}(\mathbf{c'})}{\sqrt{f(c)}} \tag{3.3}$$

The significance score $sig(c)$ for an n-gram is given by:

$$sig(c) = min_{\ \mathbf{c}'} \ \ std\_dev(c, \mathbf{c}') \tag{3.4}$$

The **ContrastSpot** algorithm to extract the contrast features for the list of potential entity variations $\mathcal{L}$ is given in Algorithm 3. First, for each pair of entities in $\mathcal{L}$, we extract all n-grams of size up to m, that are present in the unstructured attribute of one of the entities but not the other. We compute the frequency of

Algorithm 3 ContrastSpot: Extracting contrast features from potential variations

```
Input: List of variations $\mathcal{L}$, Unstructured attribute $a_v$, significance score threshold $\alpha$, max. length $m$
Output: Contrast features with weights $\mathbf{f}$
  # Compute n-gram frequencies
  for $(r_i, r_j) \in \mathcal{L}$ do
      Extract n-grams $c$ in $r_i(a_v) \triangle r_j(a_v)$
      $Freq[c] \leftarrow Freq[c] + 1$
  # Compute the set of significant phrases
  for $c \in$ Freq with length $> 1$ do
      if $sig(c) > \alpha$ then
          $Sig = Sig \cup c$
  # Extract the longest significant phrase
  for $(r_i, r_j) \in \mathcal{L}$ do
      $G \leftarrow$ Unigrams present in $r_i(a_v) \triangle r_j(a_v)$
      Grow unigrams in $G$ to largest $c \in Sig$
      $\mathbf{f} \leftarrow c$
  return $\mathbf{f}$
```

all extracted n-grams from the set of pairs in $\mathcal{L}$. Next, for all n-grams other than unigrams, we compute the significant score, and keep track of all phrases the have a score greater than the threshold α. This is the list of significant phrases.

3.3.5 Contrastive Entity Linkage

We now describe our overall contrastive entity linkage algorithm (**CEL**) for the task of three-way linkage. First, we extract the set of contrast features from both the catalogs using the **VarSpot** algorithm. Then, for each entity in the catalog, we extract phrases corresponding to contrast features from the unstructured text attributes, such as title. These phrases correspond to the set of variational attribute values. We add these phrases as an additional record attribute. We train a multiclass classifier using the labeled data to classify a pair of records as either duplicate, variation or distinct. To identify the set of duplicates $\mathcal{M}$, and the set of variations $\mathcal{V}$, we first generate the set of potential duplicate and variation pairs from the two catalogs using blocking techniques such as locality sensitive hashing (LSH). We classify these blocked pairs using the trained classifier, and return sets $\mathcal{M}$ and $\mathcal{V}$.

3.4 Experimental Validation

In this section, we perform experimental evaluation to answer the following research questions:

- *RQ1:* Do the extracted contrast features capture variational attributes?
- *RQ2:* How does contrastive entity linkage perform on the task of identifying variations and duplicates?
- *RQ3:* Does adding contrast features improve the performance of traditional entity linkage frameworks when identifying duplicates?

3.4.1 Data

We perform experiments on data of varying sizes from three different product domains: *software, music* and *groceries.* The dataset statistics are given in Table 3.3.

Software is a benchmark e-commerce entity linkage dataset that contains software products extracted from two websites, Amazon and Google [130]. Each product is associated with three attributes: title, manufacturer and price. We use the same set of blocked pairs as used in Konda et al. [128]. This is a small-sized dataset with a few thousand entities in each of the catalogs.

Groceries contains grocery products sampled from Amazon and products contained in the Open Grocery Database[1]. Each product is associated with two attributes: title and brand. To generate the set of candidate pairs, we performed blocking using locality sensitive hashing (LSH). We used both the attributes for blocking. This is a medium-sized dataset with ∼1 million entities in the Amazon catalog and ∼100,000 entities in the Opengrocery catalog.

Music is a dataset containing music tracks extracted from two different music catalogs, Musicbrainz[232] and Lastfm[160]. Each track is associated with two attributes: title and artist. To generate the set of candidate pairs, we performed blocking using LSH. We used both the attributes for blocking. This is a large-sized dataset with ∼1.5 million entities in the Musizbrains catalog and ∼1 million entities in the Lastfm catalog.

To train and evaluate the various approaches, we generated a stratified sample of the blocked pairs [23] using the Jaccard similarity computed on the two text attributes, and labeled the pairs as duplicates, variations, or distinct. The label distribution for each dataset is given in Table 3.3. Using the labeled samples, we performed Monte Carlo cross validation, and generated 10 splits by randomly splitting into train and test splits. We used 70% of the samples for training and use the remaining for testing.

3.4.2 Approaches

In our first set of experiments, we compare the performance of CEL with Magellan [128], a state-of-the-art entity linkage framework. In addition, we further investigate the utility of the **VarSpot** algorithm for identifying duplicates

[1]http://www.grocery.com/open-grocery-database-project/

by integrating the contrast features it discovers into three state-of-the-art representative, SILK [113], Magellan and Deepmatcher [168]. We first extract phrases corresponding contrast features from the title and add it as a separate attribute. For each of the frameworks, we train a model using the augmented data, and compare its performance with a model trained on data without contrast features.

Magellan is an another representative, state-of-the-art, end-to-end supervised entity linkage framework. The framework automatically generates several similarity and distance measures for each of the attributes based on the attribute type. Using these measures as features, it trains supervised classifiers such as logistic regression and random forest to perform linkage.

SILK is a representative, state-of-the-art entity linkage framework that takes as input two catalogs to link and a configuration file, and outputs the set of linked entities. The configuration file contains the set of attributes to use, similarity measures to compute on these attributes and weights for each attributes. The weights are tuned by performing a grid search using a labeled dataset.

Deepmatcher is a state-of-the-art deep learning entity linkage framework. It uses word embeddings to generate embeddings of attributes and computes a similarity representation for each of these attributes. Then a classifier is trained on the similarity representation to identify duplicates. Deepmatcher provides four different models of increasing model complexity: *SIF, RNN, Attention* and *Hybrid*. The *hybrid* model uses a bidirectional RNN with decomposable attention and a vector concatenation augmented with element-wise absolute difference to learn a similarity representation. We use the *hybrid* model as it has the highest representational power and the best performance.

Domain	**Catalog 1**	**Catalog 2**	**Blocked pairs**	**Sampled Duplicates**	**Sampled Variations**	**Sampled Distinct**
Software	1364	3227	11461	516	564	604
Groceries	1125952	110437	655254	598	1215	1412
Music	1456963	943335	1797549	412	271	1919

Table 3.3: Data statistics: Number of entities in each catalog, blocked pairs and label distribution for the three domains.

3.4.3 Performance metrics

We compute the F1 score and average precision score (APS) for each of the splits and report the mean and standard deviation. We use the Python *scikit-learn* library to compute the metrics. We performed paired t-test and numbers in bold are statistically significant with $p < 0.05$.

3.4.4 Experimental Results

Effectiveness of contrast features: We first provide a qualitative analysis of the extracted contrast features and their effectiveness in capturing the variational attributes. Recall that the first step, before performing contrastive entity linkage, is to extract the contrast features. To do this, we first block on one of the base attributes and compute a similarity score such as Jaccard on the unstructured attribute. We use manufacturer for software, brand for groceries and artist for music as base attributes for blocking. We prune blocks greater than 25 for music (large dataset), 50 for groceries (medium dataset), and 100 for software (small dataset). We did not observe considerable change in the performance when we varied these thresholds. We then compute Jaccard similarity between the title attribute of pairs and output all pairs with score greater than 0.6.

Table 3.4 shows some sample pairs discovered by the **Link** algorithm for the three domains. We observe that these entity pairs indeed correspond to variations.

Software
peachtree by sage premium accounting for nonprofits 2007
peachtree by sage premium accounting 2007 accountants ' edition
peachtree by sage pro accounting 2007
peachtree by sage complete accounting 2007

Groceries
milk duds candy 1.85 ounce boxes pack of 24
milk duds candy 5 ounce boxes pack of 3
milk duds movie size 5 oz 12 count
milk duds chocolate and caramel candy 5 ounce

Music
groove is in the heart
groove is in the heart club version
groove is in the heart sampladelic remix
groove is in the heart original version

Table 3.4: Identifying entity variations: Examples of the variations of entities linked by **VarSpot** algorithm for software, groceries and music domains. In these examples, edition is different for software, flavor and pack size is different for groceries, and versions are different for music.

While editions differ for software and pack size and flavor differ for groceries, for music the entities differ on versions.

The second step of **VarSpot** identifies contrast features and aggregates them over all pairs from the previous step. **VarSpot** identifies them by looking for phrases present in only one of the entities in a linked pair. We extract phrases up to length 3 and set α, the significance score threshold, to 3.0. That is, we consider n-grams to be phrases if their frequency is at least three standard deviations away from their expected frequency from sub-phrases. We consider the top 100 contrast features by weight for all experiments.

Table 3.5 shows some of the top contrast features by weight for different domains. We observe that edition and platform are important for software, package size and flavor are an important for groceries. For music the top contrast features correspond to different versions of the track.

Software	**Groceries**	**Music**
standard mac upgrade	pack of 6	remix
small box	pack of 2	mix
premium upsell mac	pack of 3	radio edit
standard upsell mac	2 pack	live
deluxe	original	club mix
pro	red	instrumental
upgrade	strawberry	original version
professional	orange	extended mix
mac	lemon	acoustic
home	premium	part 2

Table 3.5: Extracted contrast features: Extracted contrast features for the three domains. Most correspond to variational attributes.

	Correct	**Partial**	**Incorrect**
Contrast Features	**482**	297	436
Frequent phrases	396	455	364
Infrequent phrases	0	114	1101

Table 3.6: Interpretability of contrast features: We observe that contrast features explain about 40% of variations correctly. Frequent phrases only explain variations partially.

To evaluate RQ1 further, we ran annotation tasks using Amazon Mechanical Turk. We provide each worker with a pair of entities that are variations of each other. Along with the entities, we also provide the set of phrases present in the title of one entity but not the other. The worker annotates sets of phrases as correct if they fully explain the reason for products being variations of each other. If not all phrases have been extracted, or only part of the phrase has been extracted, the worker annotates the set of phrases as partial. If the incorrect phrase or no phrase is extracted, the worker annotates it as incorrect.

We ran the annotation task for the groceries data set as it has the largest set of variations. To show that variational attribute values cannot be captured using just the frequency of phrases, we ran the task with a set of 100 most frequent phrases and 100 most infrequent phrases. These phrases were extracted using the

			Logistic Regression		Random Forests		LR Improv.	RF Improv.
			NoCF	**CEL**	**NoCF**	**CEL**		
Software	Duplicates	F1	0.753	**0.784**	0.785	**0.81**	3.1 %	2.5 %
		APS	0.832	**0.864**	0.877	**0.897**	3.2%	2 %
	Variations	F1	0.425	**0.535**	0.677	**0.695**	11 %	1.8 %
		APS	0.56	**0.633**	0.761	**0.777**	7.3 %	1.6 %
Grocery	Duplicates	F1	0.658	**0.706**	0.717	**0.741**	4.8 %	2.4 %
		APS	0.72	**0.767**	0.809	**0.835**	4.7 %	2.6 %
	Variations	F1	0.736	0.737	0.778	**0.792**	0.1 %	1.4 %
		APS	0.761	**0.789**	0.855	**0.868**	2.8 %	1.3 %
Music	Duplicates	F1	0.665	**0.703**	0.781	0.793	3.8 %	1.2 %
		APS	0.747	**0.782**	0.854	**0.869**	3.5 %	1.5 %
	Variations	F1	0.663	**0.74**	0.765	**0.787**	7.7 %	2.2 %
		APS	0.709	**0.803**	0.838	**0.887**	9.4 %	4.9 %

Table 3.7: Contrast features improve performance: Models with contrast features (**CEL**) significantly outperform models without contrast features (**Magellan**) for both tasks across domains.

same significance score tests, but by using catalog-wide n-gram frequencies. The results of the experiment are shown in Table 3.6. We observe that just the top 100 contrast features correctly explain about 40% of the variations correctly. This is 20% more than those explained by frequent phrases. Further, another 20% of the variations are explained partially by the top 100 contrast features. Frequent phrases explain a large number of variations partially. This is because phrases such as *of 6* and *natural* are also frequent. Infrequent phrases perform poorly, as they correspond to rare product-specific attributes.

Performance of CEL: We evaluate research question RQ2 by training logistic regression and random forest models in Magellan. For the random forest classifier, we set the depth of the tree to 15 and number of trees to 1000. The mean F1 and APS scores, along with the standard deviation, for the **CEL** and **Magellan** models are given in Table 3.7. We observe **CEL** significantly outperforms models without contrast features (**noCF**) on all three domains for both duplicate and variation detection. For duplicate detection, the performance boost is up to 4.8% F1 Score and 4.7% APS (logistic regression). For variation detection, the performance boost is up to 11% F1 Score and 7.3% APS (logistic regression).

		SILK		Logistic Regression		Random Forests		Deepmatcher	
		No CF	**CF**	**No CF**	**CF**	**No CF**	**CF**	**No CF**	**CF**
Software	F1	0.702	**0.747**	0.743	**0.785**	0.785	**0.808**	0.721	0.744
	APS			0.843	**0.88**	0.878	**0.897**	0.749	0.779
Grocery	F1	0.614	**0.629**	0.681	**0.722**	0.708	**0.735**	0.647	0.674
	APS			0.725	**0.771**	0.805	**0.831**	0.664	**0.706**
Music	F1	0.572	**0.617**	0.657	**0.704**	0.771	**0.782**	0.754	0.753
	APS			0.748	**0.774**	0.848	**0.866**	0.796	0.804

Table 3.8: F1 score and APS for the task of identifying duplicates. We observe that models with contrast features (CF) outperform models without contrast features across domains.

Among the domains, we see significant improvements in the metrics for identifying duplicates for the groceries domain. This is because a large number of variations, which were getting linked as duplicates in **NoCF**, are prevented by the extracted contrast feature attribute.

Performance on duplicate detection: To evalute RQ3, we train models using the attributes present in the dataset (**NoCF** models) and compare it with models that include contrast features as separate attribute (**CF** models). We train models using Deepmatcher, Magellan and SILK. Since SILK returns the set of linked entities without scores assigned to them, we only report the F1 score for SILK.

The mean F1 and APS scores, along with the standard deviation, for the **NoCF** and **CF** models are given in Table 3.8. As in the three-class setting, we observe that **CF** models significantly outperform **NoCF** models on all three domain. For duplicate detection, **CF** models of SILK and Magellan significantly outperform **NoCF** models by up to 4.7% F1 Score (logistic regression). For Deepmatcher, we observe that **CF** models outperform **NoCF** models. However, we also observed a large variance in the metrics across the splits. The random forest models outperform Deepmatcher on all the tasks. Similarly, logistic regression outperforms Deepmatcher for the software and grocery domains. This might be due to the small amount of training data and large number of parameters present in the Deepmatcher model. For example, for the software domain, Deepmatcher models have ∼7.1 million parameters for the **NoCF** models and ∼9.2 million parameters for the **CF** models. Mudgal et al. [168] make similar observations and show that Magellan models outperform Deepmatcher models when the number of labeled data is less that 10,000.

3.4.5 Analysis of Entity Linkage

To analyze the performance of models using contrast feature we show the confusion matrix for one of the folds in the Music dataset in Table 3.9. We observe that adding contrast features helps the random forest model identify duplicates correctly which were earlier getting classified as variations. As an example the track "harvest uptown famine downtown" in catalog 1 and the track "harvest uptown" in catalog 2 was classified as a variation by the model without contrast features. This is likely because the similarity measures between the two tracks are close to that of variations. However, this was correctly classified as a duplicate by the model with contrast features as "famine downtown" in not a variational attribute and was not extracted as a contrast feature.

Without contrast features

	Pred Distinct	Pred. Dup.	Pred. Var.
Distinct	570	8	4
Dup.	11	83	23
Var.	7	7	68

With contrast features

	Pred Distinct	Pred. Dup.	Pred. Var.
Distinct	570	8	4
Dup.	13	91	13
Var.	7	6	69

Table 3.9: Confusion matrix: Models with contrast features correctly identify duplicates which are classified as variations by models without contrast features.

3.4.6 Hyperparameter Tuning

The **VarSpot** algorithm and Contrastive entity linkage has four main hyperparameters - the significance threshold α, the bucket size λ, the similarity threshold θ and the number of top contrast features used. In our experiments we set $\alpha = 3$, $\theta = 0.6$ and considered the top 100 contrast features by weight. Further we set $\lambda = 25$ for the music domain. Fig. 3.3 shows the sensitivities of these parameters to the entity linkage and variant linkage metrics (F1, APS). We plot the metrics as we vary the parameters individually. We observe that **VarSpot** algorithm and Contrastive entity linkage is robust to changes in these parameters.

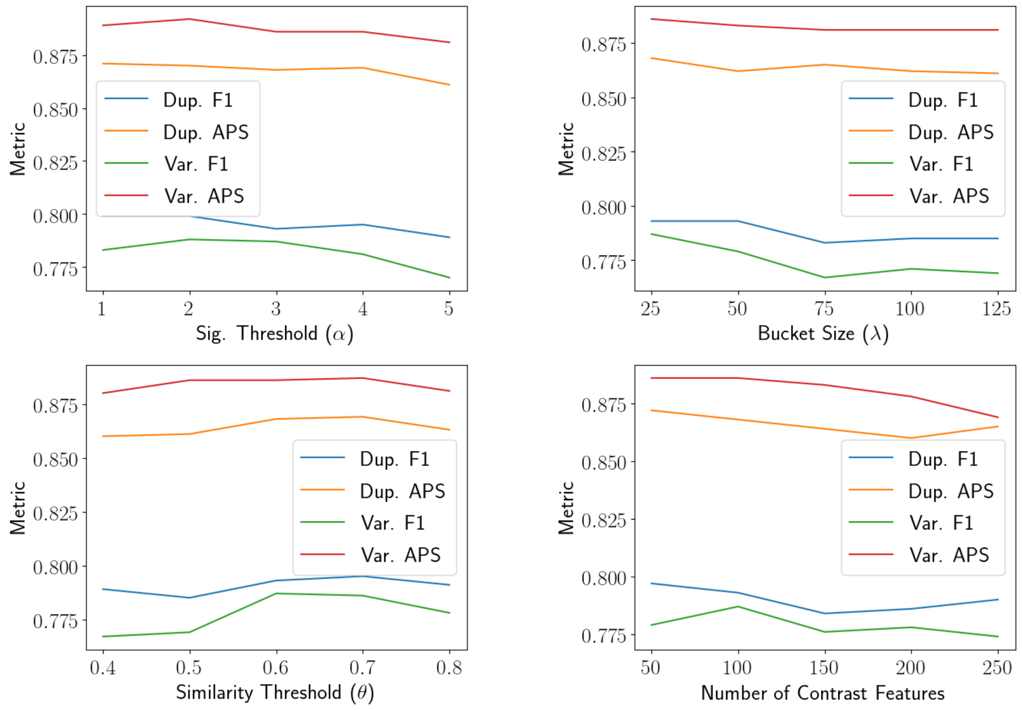


Figure 3.3: Sensitivity Analysis: Plot of entity linkage and variant linkage metrics as we vary the hyperparameters. Our approach is robust to changes in these hyperparameters.

3.5 Related Work

There is a wide body of research, spanning several decades, on the task of entity linkage [10, 54, 21, 57, 211, 53, 168, 238]. See [68, 129, 52, 132, 88, 43] for survey papers comparing various approaches. Starting from the seminal paper by Fellugi and Sunter [75], several rule-based approaches [74, 223], learning approaches [24, 128, 225] and crowdsourcing approaches [93, 245, 228] have been proposed. Recently, approaches that use deep learning models have been proposed [168, 63].

Here we review recent work in entity linkage on products, followed by approaches that identify variations of entities. Finally, we review recent work for task of attribute value extraction from text.

3.5.1 Entity linkage for products

The task of entity linkage for products has received a fair bit of attention recently [116, 131, 108, 153]. Supervised approaches such as [116, 131, 206] extract product attributes from product title and description and use these extracted attributes to perform entity linkage. Kannan et al. [116] use an inverted index to extract the set of product attribute from the product title and then use logistic regression to learn their importance. Köpcke et al. [131] use regular expressions to extract attributes and use search engines to refine them. They train a SVM using the extracted attributes to link offers. Ristoski et al. [206] use a convolutional neural network and a CRF to extract product features and train various supervised models such as random forests and SVM. Approaches such as Horch et al. [108] learn a new similarity score by combining several distance scores such as Jaccard and Sorensen distance. Londhe et al. [153] propose an unsupervised approach to link products using a community detection algorithm. They make use of a search engine to enrich the text of products. While these approaches identify duplicates, they do not identify variations.

Approaches such as Li et al. [145] identify records in a catalog that are variations of each other. They takes as input, a catalog and a partitioning of the record attributes into common-valued, dominant-valued and multivalued attributes. Using a small set of labeled data, they learn weights for the attribute values. Using these weights, they cluster the records in the catalog such that all variations are present in a cluster. Our approach on the other hand, mines variational attributes present in the unstructured text in an unsupervised manner, and uses them to identify both duplicates and variations across two different catalogs. On et al. [181] links groups of entity variations across catalogs that are the same. Our task is different, where the goal is to identify both duplicates and those that are variations of each other.

3.5.2 Attribute value extraction

The task of extracting entity attribute values from unstructured text has received significant attention as well [269, 90, 187, 148, 194, 116, 131, 206]. These approaches can be broadly classified into closed-word approaches, which assume a predefined set of attribute values [90, 148, 194], and open-world approaches which do not make such assumptions [269]. Rule-based and linguistic approaches approaches such as Chiticariu et al. [42], Nadeau and Sekine [172], Mikheev et al. [163] leverage the syntactic structure of the text to extract to attributes. CRF-based systems such as Putthividhya and Hu [194] make use of seed dictionary to bootstrap the models. Recently, neural network based models that combine LSTMs and CRF have been proposed [135, 269]. However, all these techniques extract attribute values for a pre-defined fixed set of attributes and are supervised in nature.

3.6 Conclusion and Future Work

In this chapter, we proposed our approach, contrastive entity linkage, to identify both entity pairs that are duplicates and those that are variations of each other. To address the challenge of identifying variational attributes present in the unstructured text, we proposed a scalable, unsupervised algorithm, **VarSpot**. In the experiments, using Mechanical Turk, we first showed that the contrast features are interpretable and then showed that adding contrast features as a separate attribute outperforms three state-of-the-art entity linkage systems.

This work suggests other interesting future directions. The distinction between variations and exchangeable products can often be subjective. For some consumers, the distinction between a low-sodium versus regular soup is irrelevant; for others it is highly important. An interesting direction for further work is developing and testing algorithms for personalized entity linkage. Another direction of future work is in identifying the type of product attribute captured by the contrast feature. This can enable a more fine-grained discovery of product variations such as variations that differ on flavor, variations that differ on package size and so on.

Chapter 4

Fine-grained Data Alignment in Knowledge Graphs

4.1 Introduction

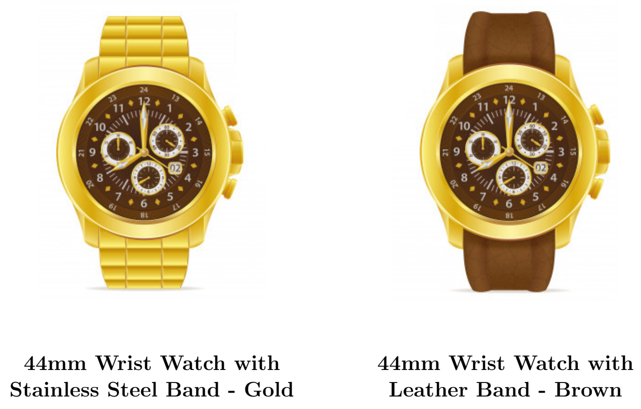

44mm Wrist Watch with Stainless Steel Band - Gold
44mm Wrist Watch with Leather Band - Brown

Figure 4.1: Product variations: Two wristwatches with different finishes and colors

How are the two products in Fig. 4.1 different from each other? While they share some of the canonical attributes, having the same manufacturer, brand and model; they have different finishes – `stainless steel` vs. `leather` and colors –

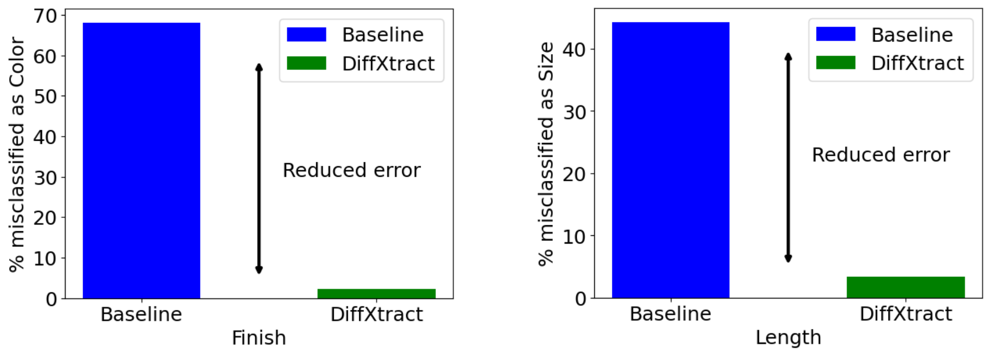


Figure 4.2: Performance for the baseline and **DiffXtract**: The baseline misclassifies finish as color for 70% of the pairs. Similarly, it misclassifies length as size for 40% of the pairs. The proposed **DiffXtract** approach identifies them accurately.

`gold` vs. `brown`. Such groups of nearly identical but distinct products are called product variations [71].

Identifying attributes that distinguish product variations is crucial for the success of a variety of online platforms. For example, e-commerce sites provide a way to select a product variation in the user interface (UI). This requires the knowledge of discriminative attributes between the variations. For conversational search agents, discovering and eliciting user's choice among the several variations is a necessary step for identifying the product the user wishes to purchase. We can also learn the preferences of a user by mining the discriminative attribute values between the products bought by the user and other available variations. This in turn significantly helps improve the user's recommendations.

Extracting discriminative attributes from product variations involves several challenges. First, each product is associated with a set of structured fields such as brand and price and a set of unstructured fields such as title and description. There are usually few structured attributes and these are often incomplete and noisy for a large set of products. Retailers highlight important attributes of the product by including them in unstructured fields such as title. For example, the information about the finish for the wrist watch bands in Fig. 4.1 (Stainless steel and leather) might not be present as a structured attribute but is mentioned in the title. While task of extracting product attribute values from unstructured text has received significant attention [257, 269, 90, 187, 148, 194, 116, 131, 206], using these approaches for discriminative attribute extraction task remains largely unexplored. Fig. 4.2 shows the performance of the baseline that extracts the discriminative values and then identifies the attribute for two attributes (more details in Section 4.4). We see that this approach incorrectly identifies color as the attribute when the true attribute is finish for about 70% of the cases. Similarly, the baseline identifies size as the attribute instead of length for about 40% of the pairs.

Second, discovering discriminative attributes involves not only extracting the attribute values but also identifying the attribute that differs between products. Different sets of products involving the same product can have different discriminative attributes. In the case of `"44mm Wrist Watch with Stainless Steel Band - Gold"` and `‘40mm Wrist Watch with Stainless Steel Band - Gold"` size is the discriminative attribute, however between the products `"44mm Wrist Watch with Stainless Steel Band - Gold"` and `"44mm Wrist Watch with Stainless Steel Band - Brown"` the discriminative attribute is color. While there is some work on extracting semantic differences between a pair of words [226, 227, 136] and extracting discriminative tokens from product reviews [121], the task of discovering discriminative attributes from product titles has not been studied.

Finally, there a large number of potential discriminative attributes. Variations in each product category vary along different sets of attributes. Often there is little or no training data for each of the possible discriminative attributes.

In this chapter, we propose a novel deep learning based approach, **DiffXtract**, to overcome these challenges. **DiffXtract** jointly identifies the discriminative attribute and extracts the attribute values for a given pair of products. The model is trained end-to-end using a multitask objective that combines attribute identification and value extraction tasks. The framework explicitly models the semantic representation of the discriminative attribute and uses attention to capture the relation between the attribute and the product title. This enables the model to scale to a large number of product attributes with little training data. From Fig. 4.2, unlike the baseline, we observe that the **DiffXtract** approach identifies finish and length attributes with high accuracy.

The main contributions include:

- **Taxonomy of approaches:** We propose a taxonomy of approaches by extending the state-of-the-art attribute identification and attribute value extraction approaches for the discriminative attribute extraction task.

- **DiffXtract:** We propose a novel end-to-end multitask approach that jointly performs both the discriminative attribute identification and value extraction tasks. We establish a theoretical upper bound for a class of approaches called extraction-oriented approaches for the attribute identification task.

- **Effectiveness:** We introduce a novel dataset and empirically show that the proposed **DiffXtract** approach outperforms attribute-oriented and extraction-oriented approaches by up to 8% F1 score when identifying attributes, and up to 10% F1 score when extracting attribute values.

To the best of out knowledge, we are the first to study the task of discriminative attribute extraction for product variations.

4.2 Preliminaries

As discussed above, variational attributes play an important role (e.g., in product browsing UI), but their values are often not provided explicitly, and are included in unstructured text fields. In this section, we first introduce necessary terms to describe the variational attributes and the formal problem definition. We then provide a discussion on existing approaches for this task.

4.2.1 Problem Definition

Product attributes can be broadly divided into two sets of attributes called *base attributes* and *variational attributes* [71]. Canonical attributes such as manufacturer, brand and product line called **base attributes**. Other product attributes that cater to the users' needs and preferences are called **variational attributes**. Color, size and quantity are some examples of variational attributes. **Product variations** are products that share the same value for all the *base attributes* but have different values for some of the *variational attributes.* For example, the two products in Fig. 4.1 are variations of each other. The variational attributes finish and color have the values `Stainless steel` and `Gold` for the first product and `Leather` and `Brown` for the second product. However, both the products share the same value, `44mm`, for the variational attribute size. We refer to the variational attribute that has different values among product variations as a *discriminative attribute.* In the example above, finish and color are the discriminative attributes.

Having defined the notion of a discriminative attribute, we now formally define the task of discriminative attribute extraction.

Definition 2. ***Discriminative attribute extraction:*** *Given a pair of product variations with titles* (T_1, T_2) *the task of discriminative attribute extraction is to output a triple* (a_d, v_1, v_2)*, where* a_d *is the discriminative attribute for the product* *pair and* (v_1, v_2) *are the extracted product attribute values for* a_d *present in the titles* (T_1, T_2).

For the pair of product variations above, the expected triple is either `{color, Gold, Brown}` or `{finish, Stainless Steel, Leather}`. We assume that the set of all variations attributes $\mathcal{A} = \{a_1, a_2, \cdots, a_m\}$ is given. However, we make an open world assumption for the attribute values. We do not assume any fixed, pre-defined vocabulary of attribute values and products can contain new emerging values that have not been seen before.

We denote the tokens in the product titles T_1, T_2 by the sets $\{t_1^1, t_2^1, \cdots, t_m^1\}$, $\{t_1^2, t_2^2, \cdots, t_n^2\}$ respectively. The extracted values v_1, v_2 are subsets of the tokens present in T_1, T_2. The set of tokens in the name of the attribute a_i (e.g., color, manufacturer part number) is represented by $\{a_1^i, a_2^i, \cdots, a_p^i\}$. If one of the product titles contains an attribute value that is missing in the other title, we do not consider this as a discriminative attribute.

4.2.2 Discriminative Attribute Extraction Approaches

Discriminative attribute extraction task involves two sub-tasks, identifying the discriminative attribute and extracting the attribute values. Based on the sub-task that is solved first, the approaches can be broadly classified into *extraction-oriented* approaches and *attribute-oriented* approaches.

Extraction-oriented approaches: These approaches extract the attribute values first and they identify the attribute that these values belong to. Since the values of the discriminative attribute needs to be different, these values must contain tokens that are present only in one of the titles. In the example of `"44mm Wrist Watch with Stainless Steel Band - Gold"` and `"40mm Wrist Watch with Stainless Steel Band - Gold"` the tokens `44mm` and `40mm` are present in only one of the titles. Since the attribute values can contain multiple tokens, and some of these tokens can be common to both the titles (e.g., `pack of 2` and `pack of 3`), these approaches start with the unique tokens and grow the values by including adjacent tokens. Once these tokens are identified, extraction oriented approaches can solve the sub-task of identifying the attribute using state-of-the-art unstructured text to product attribute matching approaches. For example, Kannan et al. [116] use a inverted index that maps values to attribute names. We could also train a multiclass classifier to identify the attribute from the tokens. The main advantage of these approaches is their ability to compare and contrast the product pairs when extracting the values.

Formally, these models can be represented by:

$$(\hat{a_d}, \hat{v_1}, \hat{v_2}) = argmax_{a_d} P(a_d | argmax_{v_1,v_2} P(v_1, v_2 | T_1, T_2)) \tag{4.1}$$

Attribute-oriented approaches: These approaches identify the attribute first and then extract the values for the identified attribute. These approaches leverage and extend the state-of-the-art product attribute extraction techniques [257, 269, 90, 187, 148, 194, 116, 131, 206]. Since the set of variational attributes is known, these approaches extract the values for each of these attributes, and then identify the attribute that has different values. In the example of `"44mm Wrist Watch with Stainless Steel Band- Gold"`, we can extract the values for all variational attribute in $\mathcal{A}$ which includes size, finish, and color. Similarly, we can extract the values for these attributes for the product `"40mm Wrist Watch with Stainless Steel Band- Gold"` Since the values for the attribute size differ we return the triplet (size, `44mm`, `40mm`). Formally, these models can be represented by:

$$(\hat{a_d}, \hat{v_1}, \hat{v_2}) = argmax_{v_1,v_2} P(v_1, v_2|T_1, T_2, argmax_{a_d} P(a_d|T_1, T_2)) \tag{4.2}$$

However, both these approaches have several drawbacks. While the attribute-oriented approaches make use of state-of-the-art product attribute extraction techniques, they extract the values for each product independently. They cannot contrast between the product pairs while extracting the values. The extraction-oriented approach, while capable of contrasting between the product pairs, cannot make use of the state-of-the-art product attribute extraction techniques as they require the attribute to be given as an input. Approaches such as Kannan et al. [116], that use an inverted index, find the attribute identification task challenging when provided with previously unseen attribute values.

More importantly, these approaches struggle to identify the correct attribute where the attributes share a large set of values. Attributes such as *color* and *finish*, for example, can take the same set of values such as *red, yellow* and *blue.* These problems are further exacerbated in attributes that take numerical values such as *length*, *width* and *height.* We can quantify the challenge of distinguishing such attributes using Bayes error rate. Bayes error rate provides an upper bound on the accuracy that can be achieved by any pattern classifier when class distributions overlap.

Let the attribute $a_i \in \mathcal{A}$ take values from the sets $\mathcal{V}^i = \{v_1^i, v_2^i, \cdots, v_m^i\}$. We denote the probability of attribute a_i taking a value v_k using the conditional distribution $P(v_k|a_i)$. In the case of categorical attributes such as *color* this takes the form of a multinomial distributions Π_i. Let the prior probability of a_i being the true discriminative attribute be denoted by $P(a_i)$.

Theorem 1. *The attribute identification task accuracy for the extraction-oriented* *approaches has an upper bound given by,*

$$1 - \sum_{v_l, v_m \in \cup_{\mathcal{V}} i} \Big(1 - argmax_i \Pi_i(v_l)\Pi_i(v_m)P(a_i)\Big)\Pi_i(v_l)\Pi_i(v_m)$$

where $P(a_i)$ is the probability of a_i being the discriminative attribute and $\Pi_i(v_m)$ is the probability of observing v_m as the value of attribute a_i.

Proof. Extraction-oriented approaches first extract the values and then identify the attributes, i.e:

$$(\hat{a_d}, \hat{v_1}, \hat{v_2}) = argmax_{a_d} P(a_d | argmax_{v_1, v_2} P(v_1, v_2 | T_1, T_2))$$

The attribute-identification task for these approaches assuming correct extraction is given by $argmax_{a_d} P(a_d | v_1, v_2)$.

From Bayes rule, the posterior probability of observing the triplet (a_i, v_l, v_m) denoted by $P(a_i | v_l, v_m)$ is given by:

$$\begin{aligned} P(a_i | v_l, v_m) &= \frac{P(v_l, v_m | a_i)}{\sum_a P(v_l, v_m | a_a) P(a_a)} \\ &\textit{Since the values } v_l, v_m \textit{are independent conditioned on } a_i \\ &= \frac{P(v_l | a_i) P(v_m | a_i) P(a_i)}{\sum_a P(v_l | a_a) P(v_m | a_a) P(a_a)} \\ &= \frac{\Pi_i(v_l)\Pi_i(v_m)P(a_i)}{\sum_a \Pi_a(v_l)\Pi_a(v_m)P(a_a)} \end{aligned}$$

The optimal Bayes classifier, assign the attribute with the highest posterior probability as the discriminative attribute i.e.:

$$\begin{aligned} \hat{a_d} &= argmax_{a_i} \frac{\Pi_i(v_l)\Pi_i(v_m)P(a_i)}{\sum_a \Pi_a(v_l)\Pi_a(v_m)P(a_a)} \\ &= argmax_i \Pi_i(v_l)\Pi_i(v_m)P(a_i) \end{aligned}$$

The classifier misclassifies when the true discriminative attribute does not have the highest posterior probability. This is called the Bayes error rate and is given by:

$$E_{bayes} = \sum_{v_l, v_m \in \cup_{\mathcal{V}} i} [1 - argmax_i \Pi_i(v_l)\Pi_i(v_m)P(a_i)]\Pi_i(v_l)\Pi_i(v_m)$$

Thus the accuracy of the classifier is upper bounded by $1 - E_{bayes}$, i.e:

$$1 - \sum_{v_l, v_m \in \cup_{\mathcal{V}} i} [1 - argmax_i \Pi_i(v_l)\Pi_i(v_m)P(a_i)]\Pi_i(v_l)\Pi_i(v_m) \tag{4.3}$$

$\square$

The accuracy upper bound decreases as the overlapping between values of different attributes increases. The upper bound is also related to the prior probabilities of the attributes. More similar the prior probabilities of attributes with overlapping values, the lower the upper bound on accuracy.

Multitask approach: To overcome these drawbacks, we propose a multitask approach that jointly performs the attribute identification task and the value extraction task. Formally, multitask approaches can be represented by:

$$(\hat{a_d}, \hat{v_1}, \hat{v_2}) = argmax_{a_d, v_1, v_2} P(v_1, v_2, a_d | T_1, T_2) \tag{4.4}$$

Multitask approaches extract both the values and identify the attribute together. These approaches look at the entire title and model the probability of attribute a_i being the discriminative attribute. The tokens in the title can potentially help these approaches circumvent the problem faced by extraction-based approaches. For example, the position of the values v_l, v_m in the tile T_1, T_2 can provide the context for identifying the attribute. Typically, the *length* tokens precede the *width* and *height* tokens. Tokens in the title that provide information about the

	Attrib. Identification	**Value Extraction**
Attribute-oriented (OpenTag[269], CAM[257], $\cdots$)		✓
Extraction-oriented (Dict [116], $\cdots$)	✓	
DiffXtract (Proposed approach)	✓	✓

Table 4.1: discriminative attribute extraction approaches: The proposed **DiffXtract** approach jointly identifies the attribute and extracts their values

product type can help distinguish between *color* and *finish*.

Table 4.1 gives an overview of the various approaches. Attribute-oriented approaches extract the values for a given attribute and Extraction-oriented approach identify the attribute given the extracted values. Our proposed multitask approach, **DiffXtract**, jointly performs both tasks and is described in the next section.

4.3 Multitask approach using DiffXtract

Our approach **DiffXtract** uses multitask learning where we train a single end-to-end model to perform multiple tasks. Our model performs all the tasks simultaneously - identifying the discriminative attribute and extracting the values for the identified attribute from each of the product titles. We cast the problem of identifying the discriminative attribute as a classification task, and the task of extracting the attribute values from the two titles as sequence labelling tasks. We identify the discriminative attribute and use this as an input for the value extraction tasks.

Our approach follows state-of-the-art neural product attribute extraction techniques and casts the attribute extraction problem as a sequence labeling task. Each token in the title is associated with a label from the set of $\{B, I, O\}$ tags,

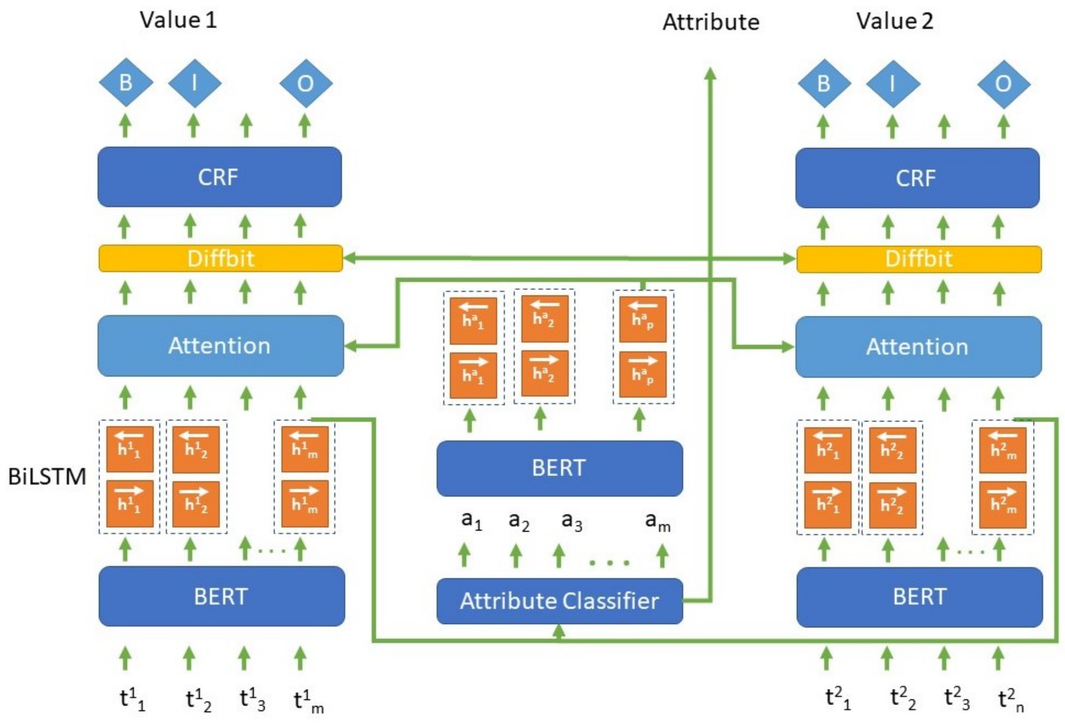


Figure 4.3: DiffXtract: Discriminative Attribute Extraction Model

where B and I denote the beginning and inside tokens of the extracted attribute value respectively, and O denotes the outside of the value tag. Neural attribute value extraction approaches use a bidirectional LSTM (BiLSTM) and conditional random field (CRF) to perform tagging. Unlike dictionary-based approaches that learn from a limited and pre-defined vocabulary of attribute values, sequence labelling approaches can generalize to previously unseen attribute values. Further, we extend the token representation to incorporate information about whether the token is unique to the product or if it is present in both the product titles. This allows the model to compare and contrast the two products when extracting the attribute values.

Fig. 4.3 shows the architecture of our proposed model. We first review the building blocks of our approach for the classification task followed by the blocks for the extraction task. We then outline our multitask strategy for the discriminative attribute extraction task.

4.3.1 Discriminative attribute identification

Word Representation Layer: Neural word embeddings map tokens in a sentence to high dimensional vectors that capture both syntactic and semantic information. We use pretrained Bidirectional Encoder Representations from Transformers (BERT) [51] to maps tokens to vectors. BERT generates contextual word embeddings by taking a sentence as input and maps each token in the sentence to a vector. Since BERT generates contextual embeddings, the embedding for the token *red* in the brand *red bull* is different from the embedding in the color *cherry red*. We add the [**CLS**] and [**SEP**] tokens to the begining and the end of the title and the generate BERT embeddings for all tokens in both the products.

Bidirectional LSTM Layer: We capture the long term dependencies between the tokens in the product titles using as Bidirectional LSTM (BiLSTM). Unlike LSTMs, which capture dependencies between a token and its preceding tokens, BiLSTMs capture dependencies in both the directions via backward and forward states. The forward and the backward hidden states are concatenated to form the final output. We feed the BERT embeddings of the tokens in the title for each of the products. We use the same BiLSTM for both the titles. The contextual representation of the titles $\mathbf{H^1}, \mathbf{H^2}$ is represented as $\{h^1_{CLS}, h^1_1, h^1_2, \cdots, h^1_m\}$ and $\{h^2_{CLS}, h^2_1, h^2_2, \cdots, h^2_n\}$ and is given by:

$$\mathbf{h^j_i} = [\overrightarrow{h^j_i}; \overleftarrow{h^j_i}] = BiLSTM(\overrightarrow{h^j_{i+1}}, \overleftarrow{h^j_{i-1}}, \mathbf{W_b}) \tag{4.5}$$

Attribute Classification Layer: We use a softmax layer to predict the discriminative attribute between the titles. We concatenate the hidden representation of [**CLS**] tokens from both the product titles and pass it to the softmax layer. The output is given by

$$P(a_d = k) = softmax([h^1_{CLS}, h^2_{CLS}].W_h) \tag{4.6}$$

where W_h is a weight matrix, h^1_{CLS} and h^2_{CLS} are the hidden representations from the BiLSTM layer of [**CLS**] tokens, and $k \in \mathcal{A}$.

Using the training data, we train the parameters of our model. We use the log likelihood or the cross entropy as the loss function as it can handle unbalanced classes in the training set. We denote the log likelihood of the classifier by l_c.

4.3.2 Attribute Value Extraction task

Having identified the discriminative attribute, we next extract the values for this attribute from each of the titles. Similar to the discriminative attribute identification task, we first encode the tokens of the title into high dimensional vectors using BERT and pass the embeddings to a BiLSTM to generate the vector representation for the tokens. Unlike the classification task, we need to generate tags for each token in the title. Hence, we use all hidden states of BiLSTM from both the titles, i.e., $\mathbf{H^1}, \mathbf{H^2}$.

Attribute representation: Similar to the tokens in the title, we generate contextual representations for all the attributes in $\mathcal{A}$. We first map the tokens present in the attribute names to high dimensional vectors using BERT and pass them to another BiLSTM and use the hidden representation of the last token as the representation of the attribute. We represent the contextual vectors for each of the attribute by h^a where $a \in \mathcal{A}$.

From the output of the attribute identification task, we select the attribute with the highest probability as the discriminative attribute and use its representation for the value extraction task. The discriminative attribute is represented as h^{a_d} and is given by:

$$h^{a_d} = h^{\hat{a}} \text{ where } \hat{a} = argmax_a P(a_d = a) \tag{4.7}$$

Attention Layer: While generating the final set of tags, the CRF considers all tokens to be equally important. However, this is not true as some tokens are more important when extracting the attribute values. In the Neural Machine Translation literature, attention mechanism was first used with great success by Bahdanau et al. [15]. Attention mechanism enables the model to *attend* to different parts of the input while generating output tags.

While generating the output tags for the tokens in the title, the attribute-based attention mechanism enables us to attend to different tokens in the title while extracting values for different attributes. This allows us to extract values for all the attributes in $\mathcal{A}$ without training a separate model for each of the attributes.

We compute the similarity between the attribute and the product title token representations to obtain attention weights $A = \{\alpha_1, \alpha_2 \cdots, \alpha_m\}$. We use cosine similarity between the attribute representation and the token representation.

$$\alpha_i = cosine(h^{a_d}, h_i^t) \tag{4.8}$$

The attribute-weighted title representation is given by $\mathbf{C^i} = A \odot \mathbf{H}^i$, where $\odot$ represents element-wise multiplication. C^i represents the weighted sum of words in the title T^i with respect to the attribute a_d.

Diffbit: Tokens that are present in one title but not the other are likely to be part of the discriminative attribute value. To provide this signal to the final layer, we compute a diffbit for each token in the title. This bit is 1 is the token is not present in the other title and set to 0 otherwise. We denote this by d^t. We append the diffbits to the token representations.

Output layer: The final output layer generates the *BIO* tags that are used to extract the attribute values. We use conditional random fields (CRF) [138] for this task as they capture dependencies in the output labels. For example, if the tag for a token is O, we know that the probability tag I for the next token is 0. We concatenate the title representations for the BiLSTM $\mathbf{H^t}$, the attribute comprehension title $\mathbf{C^t}$ and the diffbit d^t to obtain the feature matrix $\mathbf{M^t}$.

$$\mathbf{M^t} = [\mathbf{H^t}, \mathbf{C^t}, d^t] \tag{4.9}$$

This feature matrix is used by the CRF layer to generate the list of tags for each token in the title. The joint probability distribution of tags y is given by:

$$P(y_i|T;\psi) \propto \prod_{i=1}^{m} exp(\sum_{k=1}^{K} \psi_k f_k(y_{i-1}, y_i, \mathbf{M^t})) \tag{4.10}$$

where ψ_k corresponds to the feature weight, f_k is the feature function, and K is the number of features. The final output is the best label sequence y^* with the highest conditional probability, i.e.:

$$y^* = argmax_y P(y|T;\psi) \tag{4.11}$$

We learn the parameters of the model using the maximum conditional log likelihood estimate. The maximum conditional likelihood is given by:

$$l_v(\psi) = \sum_{i=1}^{N} log\ P(y_i|T_i;\psi) \tag{4.12}$$

where N denotes the number of training samples. We denote the log likelihood for the first product as l_{v^1} and for the second product as l_{v^2}.

4.3.3 Multitask Learning

Having described the building blocks of classification and the extraction model, we now describe the full model that is trained end-to-end. Fig. 4.3 shows the architecture of our proposed model. We jointly train the model for discriminative attribute identification and attribute value extraction. We do this by combining the likelihood functions l_c, l_v^1 and l_v^2. The multitask likelihood objective is given by:

$$l = l_c + \lambda(l_v^1 + l_v^2) \tag{4.13}$$

where $\lambda \in \mathcal{R}$ is a hyperparameter that trades-off between attribute identification task and attribute value extraction task.

4.4 Experimental Validation

In this section, we perform experimental evaluation to answer the following research questions

- **RQ1:** How does the proposed **DiffXtract** method compare to other techniques for the task of discriminative attribute extraction?
- **RQ2:** What attributes are harder to identify and extract?
- **RQ3:** How does the proposed **DiffXtract** method perform on the sub-task of attribute extraction?
- **RQ4:** How sensitive is the hyperparameter λ that trades-off between the two sub-tasks?

4.4.1 Data

We performed our experimental evaluation using the Multimodal Attribute Extraction dataset [152]. The dataset contains over 2.2 million products collected from several e-commerce sites. The dataset includes products from various categories such as electronics, jewelry, clothing and vehicles. Along with the product title, the dataset provides an open-schema table of attribute-value pairs. There are about 7.6 million attribute-value pairs that span across 2100 attributes.

Within an e-commerce platform, the information about which products are variations is often provided by sellers. In our evaluation, we identified product variations with a previously used approach [71]. We do this by first splitting the products into train, validation and test splits in the ratio 0.8 : 0.05 : 0.15. For each of the splits, we blocked the products using the tokens in the titles, and extracted pairs where the title Jaccard similarity is $\geq$ 0.7 and have the same value for brand. From these products, we identified product attributes that have a frequency greater than 5000. The list of identified attributes is given in Table 4.2. For these attributes, we identified pairs where each product in the pair contains the same attribute with different values among the attribute-value pairs associated with the products. For example, the products `"clear kaleidoscope static cling window film 35’ wide x 75 ft"` and `"clear kaleidoscope static cling window film 35’ wide x 82 ft"` were associated with the attribute `length` with values `75 ft` and `82 ft`. Their common attribute with different values is the ground truth for the discriminating attributes. Further, we removed pairs where the values of the discriminating attribute values were not present in the title. There were 42447 pairs in the train split, 6215 pairs in the validation split and 9910 pairs in the test split. The attribute distribution across splits is given in Table 4.2.

Attribute	Train	Validation	Test
Color	15357	3225	5316
Manufacturer part number	9203	1009	2071
Model	6062	824	970
Finish	5766	542	630
Size	1725	201	260
Length	1047	88	123
Width	975	92	128
Depth	882	50	72
Style	518	33	50
Material	254	23	42
Type	232	80	124
Dimensions	116	3	22
Height	107	6	17
Product type	91	12	21
UPC	61	3	6
Condition	21	16	56
Weight	15	3	1
Features	9	2	0
ASIN	4	0	0
Transmission	2	1	1

Table 4.2: Attribute distribution: The table show the distribution of discriminative attribute across train, validation and test splits. Here, ASIN refers to Amazon Standard Identification Number and UPC refers to Universal Product Code.

4.4.2 Approaches

We analyse the performance of extraction-oriented and attribute-oriented baselines, and compare them with the proposed **DiffXtract** model.

Dict: For the extraction-oriented approach, we evaluated an inverted index approach (**Dict**), similar to Kannan et al. [116], where we generate a dictionary containing the values that an attribute can take. We use the train and validation splits to generate this dictionary. At test time, for each of the pairs, we first identify discriminative tokens present in one of the products but not the other. For each of the products, we then identify attributes whose values are present in the title and also contain a discriminative token. We then identify common attributes that were extracted from both the products and sort them based on the sum of frequency of the attribute values. We return the top attribute and its values as the discriminative attribute.

Opentag: For the attribute-oriented approach, we compare our approach with **Opentag** [269], a recently introduced neural sequence tagging model. This approach extends the BiLSTM-CRF architecture by adding a self-attention mechanism to highlight important information before CRF layer. To ensure fair comparison with the other approaches, we used BERT word embeddings instead of the GloVe embeddings[186] proposed in the model. We use the best performing joint multi-attribute extraction model that extracts values for all possible attributes. We extract attribute-value pairs for both the titles and select the attribute with different values as the discriminative attribute value triplet. We use the training set to train the model and use the validation set to perform early stopping. We evaluate the validation set after every epoch and stop the training if the F1 score for the attribute identification task or the value extraction task decreases for three consecutive epochs.

Contextual attribute extraction model : We also extend the state-of-the-art contextual attribute extraction model (**CAM**) [257]. **CAM** is a product attribute extraction approach that takes as input a product title and an attribute, and extracts the attribute value from the product title. We extend this model by first extracting the value of all possible discriminative attributes from the title. We then identify attributes where the extracted values are different and rank the attributes based on the likelihood scores. We return the top ranked attribute along with the values as the discriminative attribute value triplet. Since this approach extracts values for each product independently, we trained the model by passing each product title in the pair and the attribute as a datapoint. We use the training set to train the model and use the validation set to perform early stopping. We evaluate the validation set after every epoch and stop the training if the F1 score for the attribute identification task or the value extraction task decreases for three consecutive epochs.

DiffXtract: This is the proposed multitask approach that jointly identifies the discriminative attribute and extracts values for that attribute. We select the attribute with the highest probability as the discriminative attribute along with the extracted values. We set parameter λ, that trades-off between the attribute identification task and attribute value extraction task to 5. We use the training set to train the model and use the validation set to perform early stopping. Similar to the above approaches we perform early stopping using the validation split.

4.4.3 Performance Metrics

We compute the precision, recall and F1 scores for the attribute identification task and the value extraction tasks. For the extraction task, we use the *Exact Match* criteria were the model gets credit only when the full sequence of extracted values are correct. As the attribute distribution is skewed, we compute the weighted macro-average where metrics are computed for each attribute and are weighted by the attribute's support.

4.4.4 RQ1: Discriminative Attribute Extraction Performance

The precision, recall and F1 scores for the attribute identification and value extraction tasks are shown in Table 4.3.

Attribute identification task: We observe that **DiffXtract** outperforms all other approaches by more than 8% on the F1 score. **CAM** performs poorly

	Attribute identification			Value extraction		
	Prec	Recall	F1	Prec	Recall	F1
Dict	0.855	0.632	0.646	0.721	0.558	0.629
Opentag	0.869	0.842	0.835	0.768	0.676	0.719
CAM	0.346	0.350	0.323	0.820	0.526	0.641
DiffXtract (No d^t)	0.853	0.875	0.858	**0.873**	0.712	0.784
DiffXtract	**0.908**	**0.912**	**0.908**	0.837	**0.751**	**0.792**

Table 4.3: Performance metrics: The table shows the precision, recall and F1 scores for the attribute identification and value extraction tasks. We observe that the **DiffXtract** approach outperforms all other approaches.

compared to all other approaches on this task. This is because the model is not trained to explicitly perform this task, and likelihood scores are not always informative with respect to attribute identification. Further, **Opentag** and **CAM** models identify attributes from each product independently and is hence unable to contrast between the pairs. **Opentag** has a separate tag for each attribute and hence is able to identify the attribute better than **CAM**. **Dict** contrasts between product pairs by identifying attribute values present in one pair but not the other. As a result, **Dict** performs better on the attribute identification task compared to **CAM**. However, **Dict** does not take into account the other tokens in a title when identifying the attribute. As a result, it performs poorly in the attribute identification task when compared to **DiffXtract**. Further, we observe that adding the diffbit (d^t) results in a performance improvement of 5% on the F1 Score.

The upper bound on the precision for the Bayes classifier mentioned in Theorem 1 on this dataset is 0.93. We note that the precision for **Dict** (0.64) is lower than the best possible value. We also observe that the precision for **DiffXtract** (0.908) is much close to this, although this approach makes use of the context in the title and can have precision higher than the Bayes classifier.

Value extraction task: We observe that **DiffXtract** outperforms all other

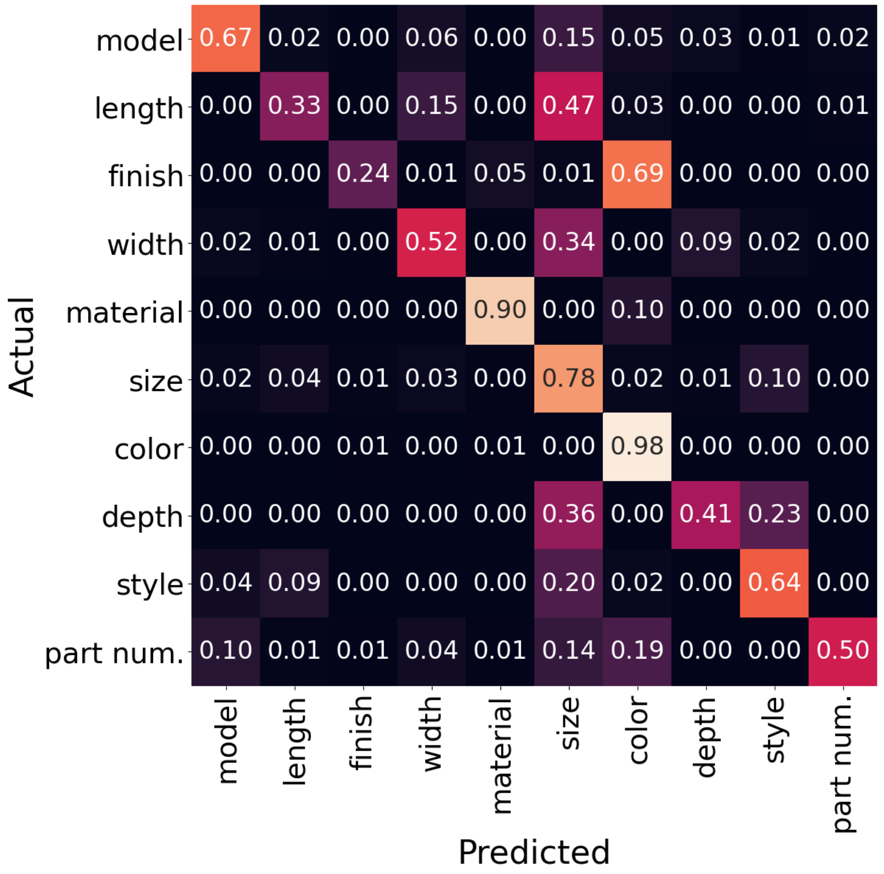


Figure 4.4: Confusion matrix for **Dict**

approaches by more than 10% on the F1 score. **Dict** performs poorly compared to all other approaches on this task. The **Dict** approach makes a closed-world assumption and cannot discover new attributes that are not present in the training data. As a result, it suffers from low recall which hurts its performance. **Opentag** does not make a closed world assumption and hence has a much higher recall. While **CAM** also does not make the closed-world assumption, the incorrect attribute identification, which is used to attend during value extraction, hurts the performance of value extraction. Like in the attribute identification task, we observe that adding the diffbit improves performance.

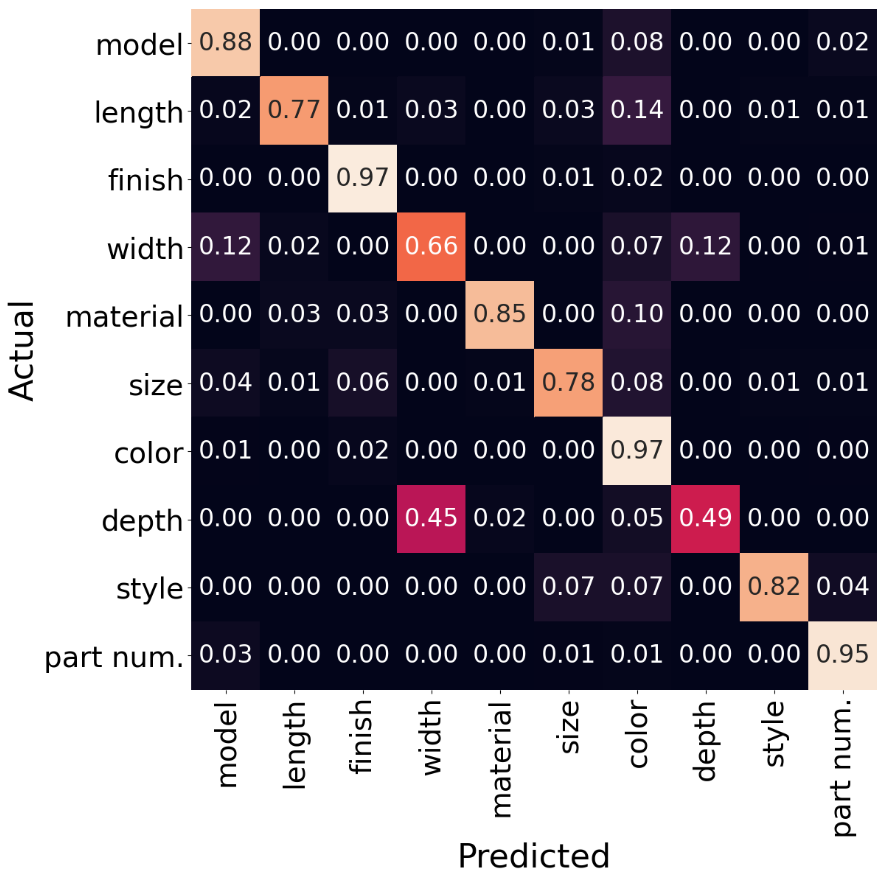


Figure 4.5: Confusion matrix for **DiffXtract**

4.4.5 RQ2: Analysis of Attribute Identification Task

To answer **RQ2**, we analyse metrics for the top 10 attributes in $\mathcal{A}$ by computing the confusion matrix. Fig. 4.4 shows the confusion matrix for the **Dict** approach. Similarly Fig. 4.5 shows confusion matrix for the **DiffXtract** approach. The diagonal elements show the fraction of test pairs for each attribute that were correctly classified, and the off-diagonal elements show the incorrectly classified pairs. We see that both approaches have high diagonal elements for some attributes suggesting that they perform the attribute identification task reasonably well for these attributes. However, for attributes that take similar values, we observe **Dict** is unable to distinguish between them. For example, the attribute

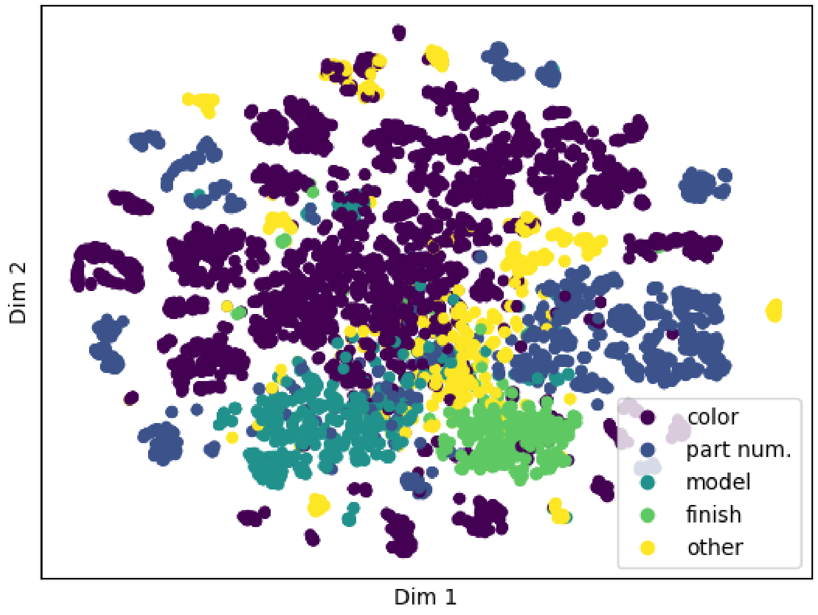


Figure 4.6: t-SNE plot: Plot shows the hidden states of [**CLS**] tokens from both the products. Each attribute forms a distinct cluster.

finish is confused for *color* in about 70% of the pairs. Similarly, the attribute for *length*, *depth* and *width* is confused for *size*. This is expected as all the above attributes are numerical in nature. **DiffXtract** model captures the token positions and context information from the other tokens in the titles. As a result, **DiffXtract** model correctly identify the attributes even when they have similar values.

To confirm this hypothesis, we plot the vector used by **DiffXtract** to identify the attribute. This contains the hidden states of [**CLS**] tokens from both the products. We project this to a two-dimensional space using t-SNE [240]. The plot for the top four attributes is shown in Fig. 4.6. We observe that variations that differ by color and those that vary by finish form two distinct clusters. The model makes use of other tokens present in the title to distinguish them. Similarly,

	Precision	Recall	F1
DiffXtract (No d^t)	0.857	0.813	0.843
DiffXtract	**0.889**	**0.839**	**0.863**

Table 4.4: Attribute extraction: Metrics for attribute extraction. We observe that providing the ground-truth improves the attribute extraction task F1 performance by 8%.

we observe variations that differ by part number and model also form different clusters.

4.4.6 RQ3: Product Attribute Extraction Task

We perform ablation experiments and evaluate the sub-task of product attribute extraction. We provided the ground-truth attribute and evaluate the value extraction performance. The extraction metrics for the the proposed **DiffXtract** model are given in Table 4.4. Comparing with the metrics in Table 4.3, we observe the F1 metric for the **DiffXtract** models improves by 8%. We observe this for both the full model and the model without the diffbit (d^t). This shows that identifying the right attribute has a significant impact on the value extraction task. The **DiffXtract** model has higher F1 score which suggests the diffbit helps in value extraction.

4.4.7 RQ4: Sensitivity to λ

To answer RQ4, we train the **DiffXtract** model with different values for λ and plot the precision, recall and F1 metrics for the two sub-tasks. The different metrics for varying values of λ are shown in Fig. 4.7. The model is robust to different values of λ. Varying the values led to slightly different epochs before the model converged.

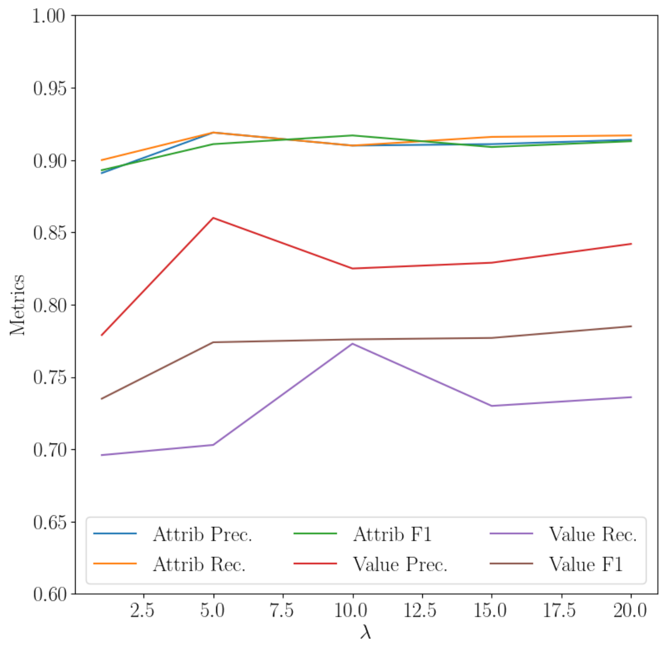


Figure 4.7: Sensitivity to λ: The model is robust to different values of λ.

4.5 Related Work

In this section we first review the related work in product attribute extraction followed by discriminative attribute extraction.

4.5.1 Product attribute extraction

The task of extracting entity attribute values from unstructured text has received significant attention [257, 269, 90, 187, 148, 194, 116, 131, 206]. These approaches can be broadly classified into closed-word approaches, which assume a predefined set of attribute values [90, 148, 194], and open-world approaches which do not make such assumptions [269, 117, 257]. Rule-based and linguistic approaches such as Chiticariu et al. [42], Nadeau and Sekine [172], Mikheev et al. [163] leverage the syntactic structure of the text to extract the attributes. CRF-based systems such as Putthividhya and Hu [194] make use of a seed dictionary to bootstrap the models. Recently, neural network based models that combine LSTMs and CRF have been proposed [135, 269, 117, 257]. Wang et al. [247] propose a multitask question answering model that casts the attribute extraction task as an answer span identification task. These approaches extract the value for a given attribute from a product title. Our task involves contrasting between two product variations and extracting both the discriminative attribute and its values.

4.5.2 Discriminative attribute extraction

McRae et al. [161] was one of the early works that studied the task of identifying important features of living and non-living entities. They introduced the notion of semantic feature production norms or McRae norms to test theories of semantic representation and computation. They collected a list of features that people think are important for set of 541 living and non-living concepts. Sommerauer and Fokkens [226] proposed an approach for investigating the nature of semantic information captured by word embeddings. Stepanjans and Freitas [227] proposed a model to explicitly detect and explain discriminative attributes from word embeddings. Krebs et al. [136] first proposed the task of semantic difference detection where the goal is to predict whether a given word could discriminate between two words. They model the semantic difference as a ternary relation between two concepts such as *apple* and *banana* and a discriminative attribute such as *red* that characterizes the first concept but not the other. Kim and Kang [121] proposed a Latent Dirichlet Allocation based approach to extract discriminative attributes from product reviews. Building on this literature, we propose the task of discriminative attribute extraction for product variations where when given a pair of product titles, the goal is to identify the discriminative attribute and extract the values from the titles.

4.6 Conclusion and Future Work

In this chapter, we proposed the novel task of discriminative attribute extraction that is crucial for product search engines and voice-based shopping assistants. We showed that approaches for this task can be classified into three groups, extraction-oriented approaches, attribute-oriented approaches and multitask approaches. We then proposed a novel multitask approach that jointly identifies and extracts the attributes. For extraction-oriented approaches, we proved a theoretical upper bound for the attribute identification task based on Bayes classifier. Empirical results on the real-world product dataset show that the proposed multitask approach outperforms all other approaches.

This work suggests other interesting future directions. The proposed approach can be extended to learn the preferences of a user by identifying the discriminative attribute values between the products bought by the user and other available variations. Generating such user profiles and using them to improve the recommendations is an interesting direction. Another direction of future work is to use the proposed approach to build a conversational search agent that can elicit user's preference and identify the right variation the user intends to buy.

Chapter 5

Aggregate Graph Queries in Knowledge Graphs

5.1 Introduction

Large realworld graphs in domains such as social media (e.g., friendship and follower graphs), computational biology (e.g., protein interaction networks), and IoT (e.g., sensor networks) often have missing information that needs to be inferred. Making use of the graph, or relational structure, can help immensely in accurately inferring missing values [217, 175]. Statistical relational learning [89, 200] and graph neural networks [92, 99, 122, 241, 197] are two powerful machine learning approaches for inferring the missing node labels. These approaches have been shown to be quite effective; however, current literature has largely focused on maximizing *locally decomposable* metrics such as node label accuracy over individual nodes.

Unfortunately, good performance on these locally decomposable metrics does not necessarily translate to accurate estimation of global graph properties. Properties such as node centrality are important in the analysis of graph phenomena such as influence maximization and resilience to attacks, and involve all the nodes and edges in the graph. Global graph properties can be computed using complex graph queries. While many such graph properties have been proposed [216, 250, 45, 201], along with efficient algorithms to estimate them [221, 151, 254, 198, 62], the task of estimating these queries when there is missing information, such as node labels, has not received much attention. In such graphs, we need to combine the tasks of estimating the queries with the inference of missing information such as node labels. These complex queries generally involve many nodes and edges and require joint reasoning over multiple node labels to compute them.

In this chapter, we introduce the notion of *aggregate graph queries* (AGQs), and argue that researchers should focus more attention on accurately estimating these richer queries. In order to support this, we introduce a suite of useful AGQs that measure subgroup cohesion in graphs [250]. We study the effectiveness of SRL and GNN-based approaches in computing AGQs on graphs with missing node labels. For approaches that infer the *best* possible values for the missing node labels, we propose a *point estimate approach*, where we first infer the missing values, and then compute the query. For approaches that infer the joint distribution over all the missing node labels, we propose an *expectation-based approach* that estimates the query as an expectation over the joint distribution. Further, to compute the expectation tractably using Monte Carlo approximation, we propose a novel sampling approach for probabilistic soft logic, one of the SRL approaches that we study.

We include a theoretical analysis that shows that the point estimate approach leads to sub-optimal estimates even for simple graphs with just two nodes. We also provide an extensive empirical analysis showing the extent to which this happens over richer queries over realworld data. Further, we analyze the effect of training data size on the performance of these approaches.

The contributions include:

- We introduce a suite of practical AGQs that measures the key graph property of subgroup cohesion and study the effectiveness of SRL and GNNs in estimating them.

- We show that first inferring the missing values and then estimating the AGQ leads to poor performance.

- We propose a novel Metropolis-within-Gibbs sampling framework, *MIG*, for PSL that is faster than existing SRL samplers.

- Through experiments on three benchmark datasets, we show that computing aggregate properties as an expectation outperforms point estimate approaches up to a factor of 50.

- The runtime experiments show that the proposed MIG approach for PSL is up to 3 times faster than other SRL sampling approaches.

5.2 Preliminaries

In this section, we recap several important statistical relational learning and graph neural network based approaches that we described in Chapter 2.

5.2.1 Statistical Relational Learning

Statistical relational learning or statistical relational learning and artificial intelligence (StarAI) methods combine probabilistic reasoning with knowledge representations that capture the structure in the domain [89, 200]. SRL frameworks typically define a declarative probabilistic model or theory consisting of weighted first-order logic rules. The rules can encode probabilistic information about the attributes and labels of nodes, and the existence of edges between nodes. Intuitively, the weight of a rule indicates how likely it is that the rule is true in the world. The higher the weight, the higher is the probability of rule being true.

SRL approaches can be broadly classified into proof-theoretic or model-theoretic approaches based on the inference technique used [50]. In proof-theoretic approaches, a sequence of logical reasoning steps or a proof is generated and this is used to define a probability distribution. Probabilistic logic programs [48] and Stochastic Logic Programs [170] are some popular proof-theoretic approaches. In a model-theoretic approach, the model is used to generate a graphical model or a ground weighted logical theory through a process called *grounding*. Inference is then performed on the ground model. Probabilistic Soft Logic [14], Markov Logic Networks [205] and Bayesian logic programs [118] are some popular model-theoretic based approaches.

Markov Logic Networks

Markov Logic Networks [205, 180, 242] are a notable model-theoretic SRL framework. A MLN induces an undirected graphical model using the set of logical rules by a process known as *grounding*. In grounding, the variables in the rules are replaced with values from the data. The atoms in the rules, where the variables are replaced with the values, are called ground atoms and are modeled as Boolean random variables(RVs) in the undirected graph. The ground rules represent cliques in the graph. Based on the data, some RVs are observed (X) and some are unobserved (Y). The probability distribution represented by the graphical model over the unobserved random variables Y is given by:

$$P(Y|X;w) = \frac{1}{Z} exp\left(\sum_{i=1}^{N} w_i \phi_i(X,Y)\right) \tag{5.1}$$

where $\phi_i(X,Y)$ is the potential defined using Boolean satisfiability, w_i is the weight, N is the number of ground formulas and Z is the normalization constant. $\phi_i(X,Y)$ takes the value 1 if the ground formula is satisfied, and 0 otherwise.

Probabilistic Soft Logic

Probabilistic Soft Logic [14] is another recently introduced SRL framework. Similar to MLNs, PSL induces an undirected graphical model using the set of logical rules. Unlike MLNs, the ground atoms in PSL are continuous and defined over the range $[0,1]$. For the potential functions, PSL uses a continuous relaxation of Boolean logic, which results in hinge functions instead of Boolean satisfiability. The probability distribution represented by the graphical model over the unobserved random variables Y is given by:

$$P(Y|X;w) = \frac{1}{Z} exp\left(-\sum_{i=1}^{N} w_i \phi_i(X,Y)\right) \tag{5.2}$$

where $\phi_i(X,Y)$ is the potential defined using Lukasiewicz logic, w_i is the weight, N is the number of ground formulas and Z is the normalization constant. The potential function $\phi_i(X,Y)$ takes the form of a hinge and makes the MAP inference in PSL convex.

5.2.2 Graph Neural Networks

GNNs build on top of neural networks to learn non-linear representation for each node in a graph. These node representations are learned by encoding information about the local graph structure [122, 241], edge labels [215], adjacent node labels [197, 188] and external domain knowledge [267, 196, 100]. GNNs can be broadly classified into non-probabilistic and probabilistic approaches based on whether they explicitly model the joint distribution.

Non-probabilistic approaches learn a non-linear representation for each node in a graph using a neural network and use them to classify nodes independently. These approaches do not explicitly model the joint probability distribution. Graph Convolutional Networks (GCNs)[122] , Relational GCN[215], Graph Attention Networks (GATs) [241] are some popular GNN approaches belonging to this category.

Recently, several probabilistic approaches have been proposed that learn a joint distribution over the unobserved node labels in a graph. The distribution is parameterized using a graph neural network. GMNN [197], ExpressGNN [267], pGAT [100], pLogicNet [196] and Column Networks [188] are some popular probabilistic approaches. To make the inference tractable, approaches such as Qu et al. [197], Qu and Tang [196], and Zhang et al. [267] use variational expectation maximization [174]. In these approaches the joint distribution is approximated with a mean-field variational distribution that is more tractable for inference. Pham et al. [188] employ an approximate, multi-step, iterative method similar to *stacked learning*, where the intermediate marginal probabilities for a node are used as relational features in the next step.

Graph Convolutional Networks

Graph Convolutional Network (GCN) [122] is a popular non-probabilistic GNN approach. GCNs iteratively update the representation of each node by combining each node's representation with its neighbors' representation. The propagation rule to update the hidden representation of a node is given by:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}H^{(l)}W^{(l)}\right) \tag{5.3}$$

where $H^{(l)}$ denotes the representation at layer l, $\tilde{D}$ represents the degree matrix, $\tilde{A}$ represents the adjacency matrix with self-loop, W represents the weights, and σ denotes an activation function, such as the ReLU. The final representations are fed into a linear softmax layer classifier for label prediction.

Graph Attention Networks

Graph Attention Networks (GATs) [241] are similar to GCNs and use self-attention while combining the representation of each node with its neighbors. This allows the model to assign different weights to each of its neighbors' representations. The propagation rule for GAT is given by:

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}} \alpha_{ij} h_i^{(l)} W\right) \tag{5.4}$$

where $h_i^{(l)}$ is the representation of node i at layer l, W is the weight matrix, $\mathcal{N}$ is the set of neighbors and α is the attention weights.

Graph Markov Neural Networks

Graph Markov Neural Networks (GMNNs) [197] is a recently introduced probabilistic approach. GMNNs build on graph neural networks such as GCNs or GATs by adding a second neural network to capture the latent dependencies in the inferred data. The pair of neural networks are trained using a variational EM algorithm. In the E-step, the object representations are learned by the first neural network. In the M-step, the latent dependencies are learned by the other neural network.

5.3 Problem Definition

Consider a graph $G = (V, \mathcal{E})$, where V is the set of nodes and $\mathcal{E}$ is the set of edges. Each node $i \in V$ is associated with a set of attributes denoted by $\mathbf{a}_i$ and a label denoted by $c_i \in \{1, \ldots, K\}$. All nodes and edges of the graph are observed and the node labels are partially observed. The set of observed node labels is denoted by C_o, unobserved node labels by C_u, and $C_o \cup C_u = C$. As an example, consider a computer science citation graph.

Example 1. *In a computer science citation graph G_c, the nodes V_c represent computer science documents and the edges E_c represent citation links between these documents. The documents in the graph can belong to several categories such as AI, Systems, Compilers and Databases. The document category is represented as a node labels C_c. The contents of the document i such as the tokens in the abstract are represent by the node attributes a_i. The documents with observed categories correspond to C_o. Documents with categories that need to be inferred are correspond to C_u.*

Definition 3 (Graph queries)**.** *A graph query GQ is a Boolean expression over nodes, edges and node labels.*

The most common form of graph queries are those that define a subgraph pattern. A graph query GQ, when evaluated on a graph G with node labels C, returns a set of subgraphs that satisfy the Boolean expression and is denoted by $GQ(G, C)$. We refer to graph queries that involve a single node or an edge as *simple graph queries*, and queries that involve multiple nodes and/or edges as *complex graph queries.*

Example 2. *For the citation graph in Example 1, we might want to infer how dense the citation links are within the categories. The GQ that returns the set of all citation links between documents that belong to the same category is given by:*

$$GQ_1 = \{\forall_{(i,j)\in V_c\times V_c}(e_{ij} \wedge c_i = c_j)\}$$

The Boolean expression is true when a pair of documents have a citation link between them and also belong to the same category.

Definition 4 (Aggregate graph queries)**.** *Aggregate graph queries (AGQs) are a class of graph queries that compute an aggregate function on the set of subgraphs that match the Boolean expression, i.e., an AGQ $Q(G, C) = Agg(GQ(G, C))$ where Agg is an aggregate function.*

For example, *Count* is an aggregate function that returns the number of subgraphs in the set. AGQs can be considered as a mapping from the graph G and the node labels C to a real number, i.e., $Q : (G, C) \to \mathbb{R}$.

Example 3. *For the citation graph in Example 1, one way to summarize the density of citations within the categories is to count the number of such citations. The aggregate graph query representing the number of citation links between documents that belong to the same category is given by:*

$$Q_1 = Count_{(i,j)}(\{\forall_{(i,j)\in V_c\times V_c}(e_{ij} \wedge c_i = c_j)\})$$

Definition 5 (Aggregate graph query estimation)**.** *Given a graph G, the observed and unobserved node labels C_o, C_u, and an aggregate graph query Q, the task of* aggregate graph query estimation *is to compute the value of $Q(G, C_o, C_u)$.*

Example 4. *For the citation graph in Example 1, the aggregate graph query in Example 3 cannot be computed directly due the missing document categories in C_u. We need to first infer the category labels before computing the AGQ.*

5.4 Aggregate Graph Queries

In this section we motivate and introduce several complex AGQs that are useful in analyzing the community structure (also called cohesive subgroups) in graphs. Analyzing the community structure of a graph is necessary to understand the social forces operating in a network and is widely used in social sciences, particularly in social psychology and sociology [250]. One of the approaches to quantitatively measure this is the nodal degree approach that computes various statistics regarding the membership of a node and its adjacent nodes to various communities.

We define five different AGQs that can be used to quantitatively measure the community structure of a graph. These queries compute statistics of the entire graph using node and edge frequencies, between nodes that belong to the same category, across different categories and also relative frequency between and across categories. We also include an AGQ that measures the accuracy of the predicted node labels to show that AGQs can also capture the traditional locally decomposable metrics. The queries are also of varying *complexity*, where the complexity is the number of nodes jointly involved in the query. Query Q0 involves a single node and queries Q1 and Q2 involve two nodes. Queries Q3 to Q5 are more complex and involve all the neighbors of a node. Q1 and Q2 are based on edge frequencies and Q3 to Q5 are based on node label frequency. We illustrate these queries using the citation graph introduced in Example 1.

[Q0]: Accuracy: This query measures the number of documents with the correct categories assigned to them. It is a locally decomposable query and is given by:

$$Q0 = Count_{c_i}(c_i = c_i^*)$$

where c_i^* is the ground truth label.

[Q1]: Edge Cohesion: This query measures the number of citation links between documents i, j that belong to the same category. It is given by:

$$Q1 = Count_{e_{ij}}(e_{ij} \in \mathcal{E} \wedge c_i = c_j)$$

A citation graph with a small number of large, tight-knit categories tends to have a large number of citations between documents of the same categories.

[Q2]: Edge Separation: This query measures the number of citation links between documents i, j that belong to the different category. It is given by:

$$Q2 = Count_{e_{ij}}(e_{ij} \in \mathcal{E} \wedge c_i \neq c_j)$$

A citation graph with large number of small communities tends to have a large number of citations between documents across different categories.

[Q3]: Diversity of Influence: This query measures the number of documents i in the graph that are connected to at least half of the different document categories. It is given by:

$$Q3 = Count_i\Big(Count_j\Big(\{c_j \mid \forall_{j=1}^{n}\Big(e_{ij} \wedge c_i \neq c_j)\}\Big) \geq \frac{K}{2}\Big)$$

The inner *Count* computes the number of distinct document categories that a document i is cited by and the outer *Count* computes the number of documents that are connected to at least half of the document categories. This query is computes the number of k-core nodes in the graph where k is set to half of the number of categories.

[Q4]: Exterior Documents: This query measures the number of documents i that have more than half of its neighbors belonging to categories other than the documents category, i.e.,

$$Q4 = Count_i\Big(\Big(Count_j\Big(\{c_j \mid e_{ij} \in \mathcal{E} \wedge c_i \neq c_j\}\Big)\Big) > \frac{Count_j(\{c_j|e_{ij} \in \mathcal{E}\})}{2}\Big)$$

The inner counts compute the number of adjacent documents with different labels and adjacent documents respectively, and the outer count computes the number of such documents in the graph. This AGQ helps measure the monophily in the graph as given by [41].

[Q5]: Interior Documents: This query measures the number of documents i that have more that half of its neighbors belonging to the same category as the document. It is given by:

$$Q5 = Count_i\Big(\Big(Count_j\Big(\{c_j \mid e_{ij} \in \mathcal{E} \wedge c_i = c_j\}\Big)\Big) > \frac{Count_j(\{c_j|e_{ij} \in \mathcal{E}\})}{2}\Big)$$

Similar to the previous query, the inner counts compute the number of adjacent documents with the same label and adjacent documents respectively and the outer count computes the number of such documents in the graphs.

5.5 Estimating Aggregate Graph Queries

In this section, we first introduce the point-estimate approach to estimate the AGQs. For models that explicitly learn the joint distribution, we also propose an expectation-based approach.

5.5.1 Point Estimation Approach

One approach for aggregate graph query estimation is to impute the *locally best* possible value for the unobserved node labels C_u and then compute the AGQ. Here, we first learn a model by minimizing a locally decomposable objective function, such as the likelihood of node labels or a loss function defined over the labels, using the graph G, node attributes $\mathbf{a}_i$ and observed node labels C_o, and impute values for the unobserved node labels C_u using the learned model. We refer to this approach as a *point estimate approach.* The point estimate approach is formally defined as follows:

Definition 6 (Point estimate approach)**.** *Given an aggregate graph query estimation task, the point estimate approach estimates Q by first imputing the values for C_u (denoted by $\hat{C}_u$) and then computes a value for Q, i.e., estimate $\hat{Q} = Q(G, C_o, \hat{C}_u)$.*

Non-probabilistic GNN approaches such as GCNs and GATs model the marginal distribution for each unobserved node label and impute labels using the mode of the distribution. SRL approaches such as PSL and MLNs, and probabilistic GNN approaches such GMNNs model the joint distribution over all unobserved node labels and impute node labels using the mode or the mean of the joint distribution.

5.5.2 Expectation-Based Approach

Another approach for aggregate graph query estimation is to define a joint probability distribution over the unobserved node labels and take the expectation of the aggregate graph query Q over the joint distribution. We refer to this approach as the *expectation-based approach.* Since the range of the aggregate graph query Q is $\mathbb{R}$, the expectation is well-defined. The expectation-based approach is formally defined as follows:

Definition 7 (Expectation-based approach). *Given an aggregate graph query estimation task, the expectation-based approach estimates Q as an expectation over the joint distribution of the unobserved node labels C_u, i.e., estimate $\hat{Q} = E_{p(\hat{C}_u|G,C_o)}[Q(G, C_o, \hat{C}_u)]$.*

AGQs can be computed as an expectation using approaches that explicitly model and perform inference on the joint distribution over the unobserved node labels. Non-probabilistic GNNs such as GCN and GAT do not model the joint distribution and cannot be used to compute the expected value. SRL approaches such as PSL and MLN and probabilistic GNNs such as GMNNs and ExpressGNNs model the joint distribution explicitly. However, computing the expectation analytically for these approaches is challenging due the intractability of the integration in the expectation. The expectation can be approximated using Monte Carlo methods by sampling from the distribution.

To make the inference tractable, approaches such as GMNN and ExpressGNN replace the joint distribution with a mean-field variational distribution. The mean-field approximation breaks dependencies between the node labels in the joint distribution. As an example, Pham et al. [188] use an iterative approach to estimate the joint distribution. The final layer of the GNN estimates the marginal node label probabilities using the labels of its neighbors from the previous iteration.

Sampling from each node's marginal distribution independently or from a mean-field distribution results in samples with limited dependence between adjacent node labels. This makes computing expectation of the AGQs using Monte Carlo approximation challenging for these approaches.

5.6 Analysis of the Estimation Approaches

In the previous section, we proposed two approaches to estimate the AGQs. In this section, we analyze the two approaches by estimating the value of the AGQ introduced in Example 3 on a graph consisting of two nodes. We use stochastic block models (SBMs) [107, 32, 1] as a generative model for the graph. SBMs are a popular class of generative models used extensively in statistics, physics, and network analysis. SBMs take as input the number of nodes n, a K dimensional vector (γ), where $\gamma_k > 0$ and $\sum_{k=1}^{K} \gamma_k = 1$, representing the fraction of nodes that belong to category k, and a $K \times K$ symmetric matrix (Π) whose elements $\Pi_{k_1 k_2}$ represent the probability of edge between two nodes belonging to categories k_1, k_2. We assume that at least one of the $\Pi_{k_1 k_2}$ where $k_1 \neq k_2$ is non-zero, i.e., there is a non-zero probability of observing an edge across nodes belonging to different categories.

The SBM generative process for a graph $G = (V, \mathcal{E})$ with node labels C is:

$$c_i \sim Multinomial(\gamma) \quad \forall i \in V$$
$$e_{ij} \sim Bernoulli(\Pi_{c_i c_j}) \quad \forall i, j \in V \times V$$

Consider a graph G with two nodes i, j connected by an edge e_{ij}. The joint distribution for the node labels c_i, c_j, under the SBM, is given by:

$$p(c_i, c_j|e_{ij}) = \frac{p(c_i)p(c_j)p(e_{ij}|c_i, c_j)}{p(e_{ij})} \tag{5.5}$$

We now show that even for the simple aggregate graph query introduced earlier that counts the number of adjacent nodes belonging to the same category, the point estimate approach leads to large errors.

Theorem 2. *For a graph G generated using SBM with two nodes i, j and an edge between them, the point estimate approach cannot minimizes the expected mean squared error for the AGQ $Q = Count_{(i,j)}(\{\forall_{(i,j)\in V\times V}(e_{ij} \wedge c_i = c_j)\})$*

Proof. The expected MSE for Q is given by $E[(Q-\hat{Q})^2]$. We know that, expected MSE is minimized when $\hat{Q} = E[Q]$, i.e.,

$$argmin_{\hat{Q}} E[(Q - \hat{Q})^2] = E[Q] \tag{5.6}$$

Since the query Q takes the value 1 when both nodes i, j have the same label and 0 otherwise, the expected value for the query Q, $E[Q]$, is equal to the probability of i, j having the same node label. Thus $E[Q]$ is given by:

$$E[Q] = \sum_{k\in C} \gamma_k^2 \Pi_{kk} \tag{5.7}$$

Since $\sum_{k_1\in C}\sum_{k_2\in C} \gamma_{k_1}\gamma_{k_2}\Pi_{k_1k_2} = 1$ and at least one of the terms $\gamma_{k_1}\gamma_{k_2}\Pi_{k_1k_2} \neq 0$ when $i \neq j$, $\sum_{k\in C}\gamma_k^2\Pi_{kk}$ lies strictly between 0 and 1. Thus $0 < E[Q] < 1$.

The point estimate approach imputes labels for the nodes i, j and estimates $\hat{Q}$ to be 1 if the imputed values i, j belong to the same category and 0 otherwise. Since the point estimate approach estimates $\hat{Q}$ to be either 0 or 1, no point-estimate approach can minimize the expected MSE. $\square$

The above theorem shows that even for simple queries, the point estimate approach leads to sub-optimal estimation. We show in the empirical evaluation that this is true also for more complex queries on larger graphs. Further, from Equation 5.6, we know that an optimal estimate can be obtained using an expectation-based approach which directly computes the expectation of AGQs under the joint distribution.

5.7 Expectation-Based Approach for PSL

In the previous section, we showed that point estimate approaches do not obtain optimal estimates. Better estimates of AGQs can be obtained by computing the expectation of AGQs over the joint distribution. Computing the expectation analytically for SRL approaches may not always be possible due the intractability of the integration in the expectation. One way to overcome this problem is to use Monte Carlo methods to approximate the expectation by sampling from the distribution. The expectation can be approximated as follows:

$$Q(G, C_u, C_o) \approx \frac{1}{S} \sum_{j=1}^{S} Q(G, C_o, C_{u(j)}) \tag{5.8}$$

where S is the number of samples and $C_{u(j)}$ are samples drawn from the distribution $p(C_u|G, C_o)$.

Gibbs sampling [91] is a type of MCMC sampling approach that generates samples from the joint distribution by iteratively sampling from the conditional distribution of each random variable. For MLNs, where conditional distributions follow a binomial distribution, approaches such as MC-SAT have been proposed [190] that combine MCMC and satisfiability.

In PSL the unobserved node labels C_u are modeled as unobserved random variables $Y_{0:m}$ where m is the number of nodes with unobserved labels. The conditional distribution for a random variable $y_i \in Y$ conditioned on all other variables X, Y_{-i} is given by:

$$p(y_i|X, Y_{-i}) \propto exp\{-\sum_{r=1}^{N_i} w_r \phi_r(y_i, X, Y_{-i})\} \tag{5.9}$$

where N_i is the number of groundings in which variable y_i participates. The above distribution neither corresponds to a standard named distribution nor has a form amenable to techniques such as inversion sampling. Hence, it is non-trivial to generate samples from the conditional distributions of PSL.

To address this challenge, unlike a previous hit-and-run based sampling approach [31], we propose a simple but effective approach for sampling from the joint distribution. We overcome the challenge of sampling from the conditional by incorporating a single step of a Metropolis algorithm within the Gibbs sampler (also called Metropolis-within-Gibbs [91]). The algorithm for our proposed approach (MIG sampler) is given in Algorithm 4. For each random variable y_i, we first sample a new value y_i' from a uniform distribution $Unif(0, 1)$ and compute the acceptance ratio α given by:

$$\alpha = \frac{exp\{-\sum_{r=1}^{N_i} w_r \phi_r(Y_{0:i-1}, y_i', Y_{i+1:m}, X)\}}{exp\{-\sum_{r=1}^{N_i} w_r \phi_r(Y_{0:i-1}, y_i, Y_{i+1:m}, X)\}} \tag{5.10}$$

We then accept the new value y_i', as a sample from the conditional with a probability proportional to α. We ignore the first b samples as burn-in. Further, for faster convergence we start the sampling from the MAP state of PSL.

Algorithm 4 MIG Sampler for PSL

```
Input: Unobserved RVs $Y$, Observed RVs $X$, $N$ ground rules, # of iterations T, burn-in period $b$.
Output: Set of samples $\mathcal{S}$
  $Y^{(0)} \leftarrow argmax_Y p(Y|X)$ // Initialize $Y^{(0)}$ to MAP state
  for $t$ from 1 to $T$ do
      for $i \in 1$ to $m$ do
          $y' \sim Unif(0,1)$
          $\alpha = \frac{exp\{-\sum_{r=1}^{N_i} w_r \phi_r(Y_{1:i-1}^{(t)}, y', Y_{i+1:m}^{(t-1)}, X)\}}{exp\{-\sum_{r=1}^{N_i} w_r \phi_r(Y_{1:i-1}^{(t)}, y_i^{(t-1)}, Y_{i+1:m}^{(t-1)}, X)\}}$
          if $Unif(0,1) < \alpha$ then
              $Y_i^{(t)} = y'$ //accept with probability $\alpha$
          else
              $Y_i^{(t)} = Y_i^{(t-1)}$
      if $t > b$ then
          $\mathcal{S} = \mathcal{S} \cup Y^{(t)}$ // Consider samples after burn-in
  Return $\mathcal{S}$
```

5.8 Experimental Validation

In this section we analyze the performance of SRL and GNN-based approaches on AGQs. We answer the following research questions:

- RQ1: How does the performance of expectation-based approaches compare with point estimate approaches?
- RQ2: How does the performance vary with the amount of labeled data?
- RQ3: What is the trade-off in performance between estimating aggregate graph queries and locally decomposable evaluation metrics such as accuracy?
- RQ4: What is the runtime performance of these approaches?

5.8.1 Data

We consider three benchmark citation datasets for node classification: Cora, Pubmed and Citeseer [217]. The nodes correspond to documents, the edges corre-

Dataset	#Categories	#Nodes	#Edges	#Attributes
Cora	7	2708	5429	1433
Pubmed	3	19717	44338	500
Citeseer	6	3327	4732	3703

Table 5.1: Statistics for the three datasets: Cora, Pubmed and Citeseer.

spond to citations, the attributes correspond to words in the document, and the categories correspond to areas of research. The statistics for these datasets are given in Table 5.1. We assume all the attributes a_i and citations $\mathcal{E}$ are observed, while the categories C are only partially observed. We generate five folds consisting 500 nodes for training, 100 nodes for validation (600 observed node labels) and use the remaining as test nodes. All approaches are given access to observed node labels during training and metrics are evaluated on the test data.

5.8.2 Approaches

SRL approaches: For both MLNs and PSL, we extend the model defined in Bach et al. [14] to incorporate node attributes. We use a bag-of-words representation for the node attributes. We train a logistic regression model(LR) to predict the node labels using the bag-of-words vectors. For each node, we consider the category with the highest probability as the LR prediction. Since LR does not need early stopping, we use all the observed node labels to train the model. We set the L2 regularizer weight to 0.001.

The model contains the following rules:

$$\begin{aligned} w_1 : HasCat(A, Cat) \land Link(A, B) &\rightarrow HasCat(B, Cat) \\ w_2 : LR(A, Cat) &\rightarrow HasCat(A, Cat) \end{aligned}$$

The predicate $HasCat(A, Cat)$ is true if document A belongs to category Cat

and predicate $Link(A, B)$ is true if documents A and B have an citation link between them. The model incorporates the logistic regression predictions using the predicate $LR(A, Cat)$, which is true if LR predicts category Cat for document A. For MLNs, we include a functional constraint that prevents a document from having multiple categories set to true. For PSL, we include a highly weighted rule that states that the truth values across all categories must sum to 1 for a node. We learn the rule weights using MC-SAT for MLN and maximum likelihood estimation for PSL using training and validation data.

The different SRL-based approaches that we consider are:

- **LR**: We compute the AGQs using the predictions of logistic regression trained on the node attributes. This is a point estimate approach.

- **MLN-MAP**: This is a point estimate approach that computes the mode of the joint distribution defined by the MLN model. We use the MaxWalkSAT algorithm implemented in the Tuffy framework [180].

- **MLN-SAM**: This is an expectation-based approach that estimates the AGQ as an expectation over the distribution defined by the MLN model. We generate 1000 samples using the MC-SAT algorithm, discard the first 500 samples as burn-in samples and randomly choose 100 samples from the 500 (to ensure minimal correlation) and use Monte Carlo approximation to compute AGQs.

- **PSL-MAP**: This is a point estimate approach that computes the mode of the distribution defined by the PSL model. We use the ADMM algorithm implemented in the PSL framework [14].

- **PSL-SAM**: This is an expectation-based approach that estimates the AGQs as an expectation over the distribution defined by the PSL model. Similar to MLN-SAM, we generate 1000 samples are generated using the proposed *MIG-sampler* introduced in Algorithm 4, discard the first 500 samples as burn-in samples and randomly choose 100 samples from the 500 (to ensure minimal correlation) and use Monte Carlo approximation to compute AGQs.

GNN based approaches: These are point estimate approaches that use the node representations to infer node labels. These models are trained using the training and validation data where the validation data is used to perform early-stopping. We used the code provided by the authors of the respective papers. For all three approaches we performed hyperparameter tuning and found that hyperparameters provided by authors performed best. The different GNN-based approaches we consider are:

- **GCN**: This approach uses the representation computed using a graph convolutional network [122].
- **GAT**: This approach uses the representation computed using a graph attention network [241].
- **GMNN**: This approach uses the representation computed using a Markov neural network introduced recently [197].

5.8.3 Performance Metrics

In Subsection 5.8.4 and Subsection 5.8.5, we evaluate the performance on the AGQs (Q0 to Q5) using the relative query error (QE) and in Subsection 5.8.6 we evaluate the categorical accuracy (Acc) and homophily error. QE is computed using:

$$QE = \frac{|\hat{Q} - Q|}{Q} \tag{5.11}$$

where Q is the true value of the query and $\hat{Q}$ is the predicted value. We evaluate the overall performance of each method by computing the average QE over all queries denoted by AQE. For homophily error we use the homophily measure H defined in Dandekar et al. [46] and compute error similar to QE by computing the absolute difference w.r.t. true H computed using the true labels. Homophily measure H is given by:

$$H = \frac{|e \in S|}{|e \in NS|} = \frac{Q1}{Q2} \tag{5.12}$$

where S and NS as sets of edges such that the nodes have the same category and not the same category, respectively. All reported metrics are averaged across five folds.

5.8.4 Performance on AGQs

In this section we answer RQ1 by computing the QE for the AGQs proposed in Section 5.4. The QE and AQE for all datasets are shown in Table 5.2. We observe that PSL-SAM has the lowest or the second lowest error across most of the non-decomposable queries ($Q1-Q5$). GNNs perfrom better on accuracy ($Q0$) which is a locally decomposable query. In Citeseer, although LR performs worse on locally decomposable AGQs such a $Q0$, it performs better on other AGQs. This is due to the sparse nature of the graph, where non-collective approaches perform better. Among collective approaches, PSL-SAM outperforms all other approaches. GNNs have a high query errors for non-decomposable AGQs. This is consistent with our theoretical analysis.

Among the queries, we observe that Q1 and Q5 have lower error compared to the other queries for all the methods. Both Q1 and Q5 estimate node pairs that are adjacent and have the same category. These are easier to estimate as these nodes typically lie at the center of the category clusters. Since all the

(a) Query error for Cora

Methods	Q0	Q1	Q2	Q3	Q4	Q5	*AQE*
GCN	<u>0.143</u>	0.0756	0.323	0.281	0.768	0.363	0.325
GAT	0.159	0.076	0.326	0.281	0.729	0.361	0.322
GMNN	**0.142**	0.081	0.348	0.254	0.754	0.367	0.324
LR	0.324	0.320	1.371	1.854	0.993	0.401	0.709
MLN-MAP	0.188	**0.011**	0.110	0.136	0.529	0.268	0.207
PSL-MAP	0.162	0.027	0.116	<u>0.063</u>	<u>0.060</u>	<u>0.034</u>	0.077
MLN-SAM	0.170	0.021	<u>0.092</u>	0.068	0.074	0.035	<u>0.076</u>
PSL-SAM	0.170	<u>0.015</u>	**0.066**	**0.005**	**0.040**	**0.022**	**0.053**

(b) Query error for Pubmed

Methods	Q0	Q1	Q2	Q3	Q4	Q5	*AQE*
GCN	**0.152**	0.129	0.524	2.732	0.737	0.126	0.733
GAT	0.168	0.144	0.583	2.364	0.764	0.132	0.692
GMNN	<u>0.157</u>	0.134	0.545	2.581	0.743	0.127	0.714
LR	0.219	0.126	0.513	6.342	0.712	0.120	1.33
MLN-MAP	0.205	0.075	0.362	3.613	0.435	0.080	0.795
PSL-MAP	0.170	<u>0.016</u>	<u>0.064</u>	4.259	**0.007**	**0.001**	0.752
MLN-SAM	0.223	0.037	0.070	<u>0.057</u>	0.051	0.007	<u>0.073</u>
PSL-SAM	0.171	**0.009**	**0.038**	**0.011**	<u>0.022</u>	<u>0.003</u>	**0.042**

(c) Query error for Citeseer

Methods	Q0	Q1	Q2	Q3	Q4	Q5	*AQE*
GCN	**0.263**	0.241	0.736	0.876	0.907	0.429	0.575
GAT	0.272	0.254	0.775	0.888	0.939	0.439	0.594
GMNN	<u>0.268</u>	0.251	0.766	0.867	0.905	0.412	0.578
LR	0.327	**0.134**	**0.408**	**0.378**	**0.382**	**0.176**	**0.300**
MLN-MAP	0.292	0.192	0.595	0.625	0.789	0.369	0.477
PSL-MAP	0.283	0.151	0.460	0.600	0.516	0.237	0.374
MLN-SAM	0.297	0.161	0.506	0.641	<u>0.485</u>	<u>0.217</u>	0.384
PSL-SAM	0.286	<u>0.143</u>	<u>0.435</u>	<u>0.586</u>	0.509	0.231	<u>0.365</u>

Table 5.2: Query error obtained for all queries on the three datasets and the average query error (AQE) across queries. The lowest error is indicated in bold and the second lowest error is underlined.

approaches propagate the similarity between the node neighbors, the models have a lower error on these queries. Queries Q2, Q3, and Q4 estimate nodes that have neighbors with different categories. These are nodes that lie in the boundary of the category clusters and are harder to infer. GNN-based approaches have very large errors for these queries, resulting in overall poor performance.

5.8.5 Effect of Training Data

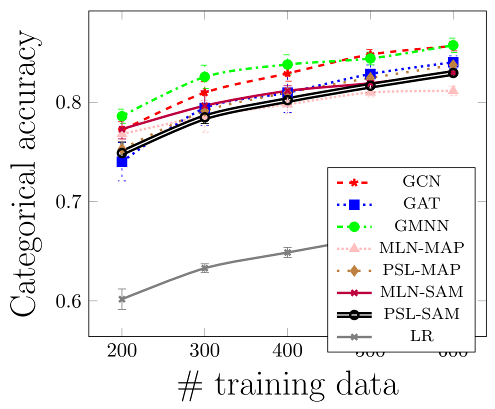
To address RQ2, we create five variants of the datasets by varying the amount of training data available for each method from 200 to 600 with increments of 100. Fig. 5.2 shows the performance of different methods on AGQs as we increase the number of training examples. We report the mean and the standard deviation of AQE across the five folds. We observe that on all three datasets expectation-based approaches have the lowest error. The average query error for logistic regression decreases sharply in the Citeseer data as we increase the size of the training data. We also observe that expectation-based approaches are more robust to the amount of training data when compared to point-estimate approaches.

5.8.6 Trade-off between Estimating AGQs and Locally Decomposable Metrics

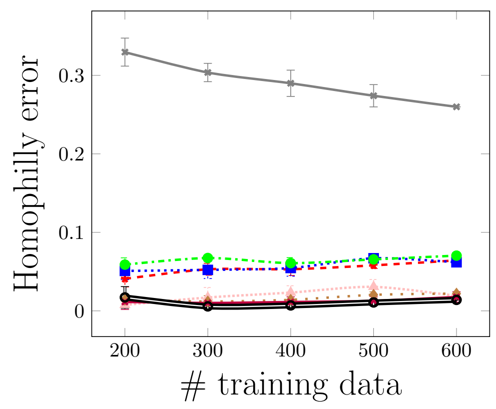
To answer RQ3, we compute the accuracy of the predicted node labels which is locally decomposable. Accuracy involves correctly estimating the node labels of each node individually. Estimating AGQs, on the other hand, requires correctly estimating the node labels for several adjacent nodes.

In Fig. 5.1, we plot the accuracy of the predicted node labels for all three datasets with different amount of training data. We observe that GNNs have a higher accuracy compared to SRL approaches. This is due to the sparsity of node attributes in these datasets which leads to inferior predictions by the logistic regression classifier. GNNs overcome this sparsity by aggregate features of the neighboring nodes. However, this implicitly assumes that a node's neighbors have

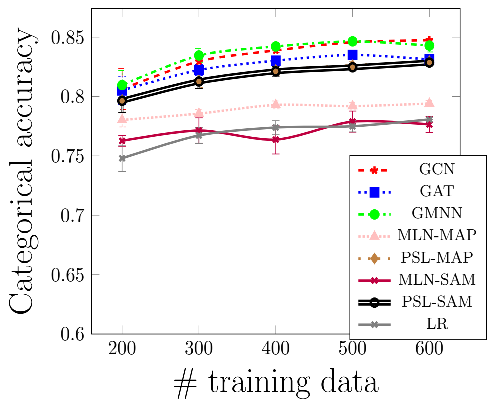
(a) Accuracy in Cora dataset.



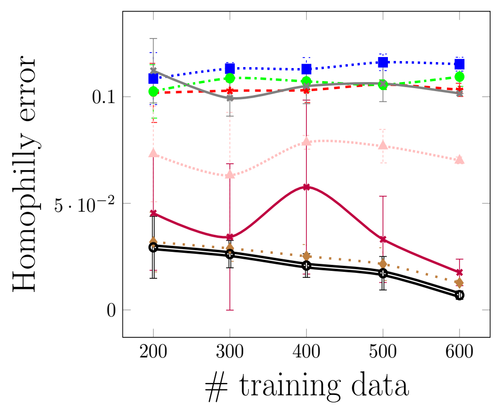
(b) Homophiliy error in Cora dataset.



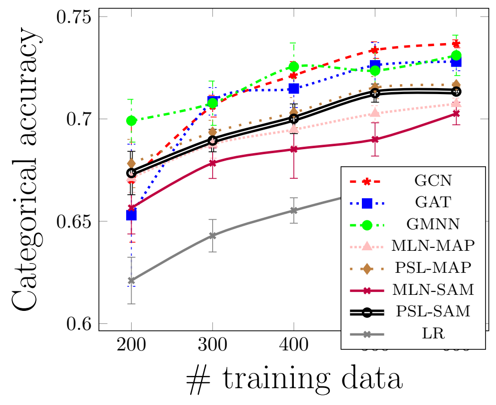
(c) Accuracy in Pubmed dataset.



(d) Homophiliy error in Pubmed dataset.



(e) Accuracy in Citeseer dataset.



(f) Homophiliy error in Citeseer dataset.

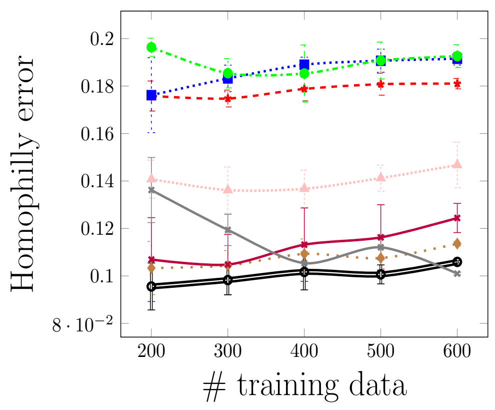


Figure 5.1: Figures shows the accuracy and homophily error of different approaches on all three datasets as the number of training data increases.

the same label. While this is true for most nodes, it is not always true. As a result, GNNs tend to perform poorly on AGQs which involve correctly estimating multiple node labels that may belong to different categories.

The error in the homophily between the estimated node labels and the true labels in shown in Fig. 5.1. We observe that GNN-based approaches have a large error when compared to SRL approaches. Further, we observed that by artificially modifying the weight of the first rule in PSL that propogates the node labels across the citation edges, the accuracy could be improved at the cost of poor AGQ estimates. This shows that there is a trade-off between locally decomposable metrics such as accuracy and AGQs. While GNN-based approaches are good at estimating locally decomposable metrics they perform poorly when estimating AGQs. SRL-based approaches due to their flexibility in modeling can be altered to perform well on either of the two metrics.

5.8.7 Runtime Comparisons

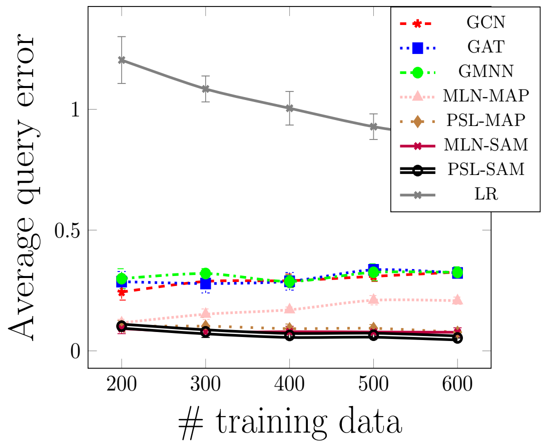
Methods	Cora Time (sec)	Pubmed Time (sec)	Citeseer Time (sec)
GCN	24	59	29
GAT	142	138	122
GMNN	30	17	8
LR	2	5	2
PSL-MAP	14	124	37
MLN-MAP	65	368	36
PSL-SAM	105	638	124
MLN-SAM	270	1947	166

Table 5.3: Table showing runtimes for each of the approaches on the three datasets.

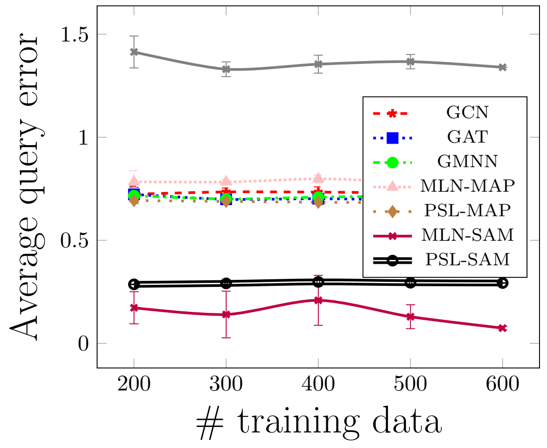
To answer RQ4, we recorded the runtimes of the different approaches and is given in Table 5.3. As expected, we observed that point estimate approaches are significantly faster compared to expectation-based approaches. This is not surprising as point estimates are computed using efficient optimization approaches. Among the GCN, GAT, GMNN, PSL-MAP and MLN-MAP, we observe that GMNN takes the least amount of time in Pubmed and Citeseer dataset and PSL-MAP takes the least amount of time in Cora dataset. Among PSL-SAM and MLN-SAM, we observe that our proposed MIG sampler for PSL is faster than MLN-SAM by a factor of two for Cora and three for Pubmed.

5.9 Conclusion and Future Work

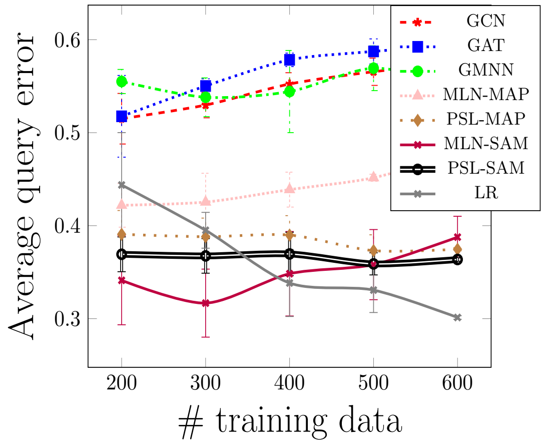
In this chapter, we motivate the practical need for aggregate graph queries (AGQs), and show that existing approaches which optimize for locally decomposable metrics such as accuracy neither perform well theoretically nor empirically. In order to compute the expectation under the joint distribution, we introduce a novel sampling approach, MIG, for PSL that is both effective and efficient. We perform an extensive evaluation of SRL and GNN approaches for answering AGQs. Through our experiments we show that SRL methods can get up to 50 times less error compared to GNNs and that our proposed MIG sampler is up to three times faster than other SRL sampling approaches. An interesting future direction is to combine GNN approaches with SRL models that can learn node representations and also infer a joint distribution over the unobserved data. Extending this analysis for networks with missing edges and nodes is another interesting line of future work.



(a) *AQE* for Cora dataset.



(b) *AQE* for Pubmed dataset.



(c) *AQE* for Citeseer dataset.

Figure 5.2: Effect of training data on average query error (*AQE*) for various approaches.

Chapter 6

Model Discovery in Knowledge Graphs

6.1 Introduction

Templated graphical models (TGMs) are a broad class of probabilistic graphical models that use templates to succinctly define the structure of a model. A template encodes the relationship between random variables and is instantiated many times within the model [127]. TGMs have shown remarkable success in capturing probabilistic dependencies in structured data and have been successfully applied to many domains including computational biology, knowledge base completion, and text mining.

Learning these templated models directly from the data, also known as *structure learning*, alleviates the cost of human-engineered model construction. However, the task of structure learning involves several critical computational challenges. First, the model space is potentially infinite and, even when restricted to be finite, results in a combinatorial search. Second, approaches that iteratively refine and grow a set of rules require interleaving of several costly rounds of parameter estimation and scoring. Finally, scoring the model often involves computing the model likelihood, which is typically intractable to evaluate exactly.

In addition to predictive performance, there is a growing interest in explainable models. Models that provide explanations lead to increased user trust and have also been shown to be more persuasive [235, 204, 9, 60, 265, 246]. Explanations can also help isolate and identify incorrect assumptions and biases learned by the model. While TGMs are more interpretable than other large graphical models, generating explanations for individual predictions that satisfy certain desired properties is still challenging. Further, not all data sources that are included in the model can be explained to the end user. When learning a model from the data, there is a need to trade-off between accuracy and end-user explainability.

In this chapter, we propose a novel approach, *explainable structured model search* (*ESMS*), that learns an explainable templated graphical model automatically from data. The data consists of several information sources, not all of which can be easily explained to the end user. Our proposed approach leverages probabilistic soft logic (PSL)[14], a TGM defined using a set of weighted first-order logic rules. Unlike other SRL methods that use Boolean logic, PSL uses Lukasiewicz logic, a continuous relaxation of Boolean logic, and can incorporate real-valued data such as similarity metrics and confidence scores. Our proposed structure learning approach utilizes an efficient weight learning strategy that invalidates the need to re-optimizing a rule's weight across models during the search. To prune non-informative rules that might be generated, we introduce an efficient learning objective for PSL that assigns importance weights for rules and eliminates non-informative rules.

Our *ESMS* approach introduces an explanability score that biases the search to learn end-user explainable models. After learning a model, we generate explanations for each of the inferred random variables. The continuous nature of the PSL inferred values enables our explanation framework to satisfy the properties of *explicitness* and *faithfulness* [9]. Further, we extend the concept of *stability* for the SRL setting and show that the proposed explanation strategy is stable.

The main contributions include:

- We propose a novel structured search approach that efficiently discovers a templated graphical model using rule templates that best captures the statistical dependencies in the data.
- We introduce an efficient weight learning strategy based on a piecewise pseudolikelihood objective that allows parallelization and requires weights for a template to be learned only once across models.
- Using an explainabilty parameter, our learning approach generates models that trade-off accuracy and end-user explainability of its predictions.
- We propose a new Fisher score based ranking algorithm that identifies the best explanation for a prediction and theoretically show that this is stable.
- We empirically show, for the recommendation domain, the discovered models using our proposed approach outperform models generated using state-of-the-art methods.

6.2 Preliminaries

In this section, we recap the probability distribution defined by Probabilistic soft logic. The templates in PSL are weighted logical clauses that encode statistical dependencies and structural constraints. Consider the following PSL rule that suggests similar items are rated similarly by the user:

$$w : \mathit{SimilarItem}(I_1, I_2) \wedge \mathit{Rating}(U, I_1)\,;\implies\,;\mathit{Rating}(U, I_2)$$

Here, *SimilarItem* is a **predicate** that encodes the similarity between the two items I_1 and I_2, and the predicate *Rating* encodes the rating assigned to the item by the user U. w denotes the weight of the rule that determines its importance. The **variables** I_1, I_2, U range over the **constants** in a domain. The number of variables in a predicate is called the **arity** of the predicate. The predicate together with the list of variables is called an **atom**. The set of predicates under consideration is denoted by $\mathcal{P}$. Given a set of users and items such as User = {Alice, Bob} and Item = {Top Gun, Valkyrie}, PSL generates **ground rules** by substituting variables in the rules with constants. An example of a ground rule is as follows:

$$\begin{aligned} w :&\mathit{SimilarItem}(Top\ Gun, Valkyrie)\wedge \\ &\mathit{Rating}(Alice, Top\ Gun)\,;\implies\,;\mathit{Rating}(Alice, Valkyrie)\end{aligned}$$

The atoms in a ground rules are called **ground atoms** (e.g. *SimilarItem*(Top Gun, Valkyrie)). Values of ground atoms maybe known (**observed ground atoms**) or need to be inferred (**unobserved ground atoms**). A **PSL Model** (denoted by $\mathbf{M}$) is a set of weighted rules $\{r_1, r_2, \cdots, r_n\}$.

Using the model $\mathbf{M}$ and a set of observed and unobserved ground atoms, PSL generates a hinge-loss Markov random field. PSL first associates a random variable in the range $[0, 1]$ with each ground atom. Random variables that correspond to ground atoms with observed values are called **observed ran-**

dom variables ($\mathbf{X}$) and those that correspond to ground atoms with unobserved values are called **unobserved random variables** ($\mathbf{Y}$). For the ground rule mentioned above, let X_1, Y_1, Y_2 be the random variables associated with the ground atoms $SimilarItem(Top\ Gun, Valkyrie)$, $Rating(Alice, Top\ Gun)$, $Rating(Alice, Valkyrie)$. Then each grounded rule is mapped to a hinge-loss potential ϕ using Lukasiewicz logic. For the ground rule mentioned above the hinge-loss potential is given by:

$$\phi(\mathbf{Y}, \mathbf{X}) = max\{X_1 + Y_1 - Y_2 - 1, 0\}^p \tag{6.1}$$

where $p \in \{1, 2\}$. In this work we consider $p = 2$, which results in squared hinge-loss potentials.

Given the set of observed and unobserved random variables $\mathbf{X}, \mathbf{Y}$, and the set of potentials Φ, PSL defines a probability distribution over the unobserved random variables $\mathbf{Y}$ conditioned on the observed random variables $\mathbf{X}$ as follows:

$$\begin{gathered} P(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z(\mathbf{X}, \mathbf{Y})} exp(-E) \\ where\ E = \sum_j w_j \Phi_j(\mathbf{Y}, \mathbf{X})\ ;\ Z(\mathbf{X}, \mathbf{Y}) = \int_{\mathbf{Y}} exp(-E) \end{gathered} \tag{6.2}$$

Here, j iterates over all the ground rules, and the w's are weights. The function E is called the **energy function**.

6.3 Explainable Templated Graphical Models

Explanations are human-understandable artifacts that provide qualitative understanding of the relationship between the data, the model's internal state, and the predictions [204, 252]. Explanations can either be generated a posteriori, where the model is viewed as a black box, or generated by the model internally along with its predictions. A good explanation must satisfy three properties -

explicitness, *faithfulness*, and *stability* [9]. Explicitness means that the generated explanation is interpretable by the user. A faithful explanation implies that the generated explanation is relevant to the prediction. Finally, stability means that the generated explanation does not change drastically for small changes in the input features.

In the relational setting, the generated explanations depend on other observed and unobserved random variables. A stable explanation must not change drastically when the values of other random variables change. We refer to this as *relational stability*. We formally define this by extending the framework in Wolf et al. [252] to a relational setting.

Let h be a model that predicts the values for the unobserved random variables ($\mathbf{Y}$) given the observed random variables $\mathbf{X}$. We denote an explanation by G, and the space of possible explanations by $\mathcal{G}$.

Definition 8. ***Explaining function:*** *An explaining function, denoted by f, scores the possible explanations faithfully and takes as input three components: the input, the output and an explanation, i.e., $f : (\mathbf{X}, \mathbf{Y}, \mathcal{G}) \rightarrow \mathbb{R}$.*

Definition 9. ***Relational Stability****: Let h be a model and f be an explaining function. Let $\mathbf{X}, \mathbf{Y}$ be the set of observed and unobserved random variables and $\mathcal{G}$ be the space of possible explanations. We say that f is stable with respect to h, if for any two $\mathbf{X_1}, \mathbf{X_2} \in \mathcal{X}$ that differ in a single random variable X_i and a prediction $Y_k \in \mathbf{Y}$, $\exists D \in \mathbb{R}$ such that:*

$$\forall G \in \mathcal{G} \ \forall k \ \ |f_k(\mathbf{X_1}, h(\mathbf{X_1}), G) - f_k(\mathbf{X_2}, h(\mathbf{X_2}), G)| \leq D \tag{6.3}$$

The above definition states that the explaining function score for a prediction's explanation does not vary a lot when the value of one of the observed random variabless is changed.

Having defined relational stability, we now define the task of learning explainable templated graphical models.

Definition 10. ***Learning explainable templated graphical models:*** *Given a set of predicates $\mathcal{P}$ along with a target predicate $\mathbf{P}_T \in \mathcal{P}$ that we need to infer. The task of learning explainable templated graphical model involves two subtasks:*
Structure Learning: *The structure learning subtask involves discovering a templated model $\mathbf{M}$ that is then used to infer the values of $\mathbf{Y}$ that belong to the predicate $\mathbf{P}_T$.*
Explanation: *The explanation subtask involves generating and ranking the explanations for the inferred values of $\mathbf{Y}$ using the explanation function f that satisfies the three properties of explicitness, faithfulness and relational stability.*

6.4 Structure Learning for Templated Graphical Models

Learning a templated graphical model directly from data poses two main challenges. First, even after restricting the template length and the size of the model, it involves a combinatorial search and the possible set of models is very large. Second, the search over the space of models involves estimating the weights of the templates many times, which is costly. To overcome these challenges, we first introduce the notion of *rule patterns* that capture the common statistical patterns and perform a structured search by sampling the space of models using these patterns. Further, we propose a novel likelihood function, piecewise-pseudologlikehood (PPLL), to learn the weights of the inferred rules.

We illustrate our approach using patterns from the recommender system domain, however, the patterns are quite general, and apply in many other domains that involve multi-relational reasoning.

Definition 11. ***Rule Patterns:*** *A rule pattern is a skeletal rule that encodes the structure and variable bindings between the predicates.*

Rule patterns capture the common statistical relational patterns present in the data. Assigning predicates to the slots in the pattern results in a template.

Consider the simplest possible pattern:

$$\underline{\quad}(A, B) \rightarrow \mathbf{P}_T(A, B)$$

where $\underline{\quad}$ is a slot that can be filled by a predicate. This pattern can be used to combine or fuse information from multiple sources. For example, it can be use to generate templates for a hybrid recommender system[133] that combines information from other simpler recommendation systems, e.g.:

$$NMF(U, I) \rightarrow Rating(U, I)$$

where NMF (Non-negative Matrix Factorization) is a standalone recommender system.

We propose three sets of rule templates that capture a wide variety of rules used in templated graphical models.

Path Template: The path template is the most common template and can capture relational patterns such as transitivity. Each slot in the template must be filled with a predicate of arity two. A path template of size two has the following structure:

$$\underline{\quad}(A, B) \wedge \underline{\quad}(B, C) \rightarrow \mathbf{P}_T(A, C)$$

An example for a path based rule in recommender systems is:

$$SimilarItem(A, B) \wedge Rating(B, C) \rightarrow Rating(A, C)$$

Similarly, path templates of size three and higher can be defined.

Local Template: The local template can incorporates information from multiple standalone sources. The local template has the following structures:

$$\begin{aligned} \_\_(A, B) &\rightarrow \mathbf{P}_T(A, B) \\ \_\_(B) &\rightarrow \mathbf{P}_T(A, B) \\ \_\_(A) &\rightarrow \mathbf{P}_T(A, B) \end{aligned}$$

In addition to our earlier hybrid recommender example, another example of a rule generated from the local template is:

$$AvgUserRating(A) \rightarrow Rating(A, B)$$

Similarity Template: The similarity template captures the relational between multiple target instances. Each slot in the template must be filled with a predicate of arity two and has the following structure:

$$\_\_(A, B) \wedge \_\_(C, A) \rightarrow P_T(C, B)$$

For example, similarity functions used in collaborative filtering can be generated from this template and is given by:

$$SimilarItem(I_1, I_2) \wedge Rating(U, I_1) \rightarrow Rating(U, I_2)$$

Algorithm 5 **Explainable Structured Model Search (*ESMS*)**

```
Input: $T$: Rule templates; $L_M$: max rules in a model; $N$: max iterations; $P$: Set
  of open and closed predicates
Output: $\mathbf{M}^*$: optimal model
  $score_{best} \leftarrow -\infty$
  $\mathbf{M}^* \leftarrow \phi$
  for $i \in 1$ to $N$ do
      $\mathbf{M} \leftarrow \phi$
      while $l_{\mathbf{M}} < L_M$ do
          $r \leftarrow generateRule(T, P)$
          $w_r \leftarrow \arg\max_{w_r} l_{ppll}(\mathbf{w})$
          $\mathbf{M} \leftarrow \mathbf{M} \cup r$
          $l_{\mathbf{M}} += 1$
      if $V(\mathbf{M}) > score_{best}$ then
          $\mathbf{M}^* \leftarrow \mathbf{M}$
          $score_{best} \leftarrow V(\mathbf{M})$
  return $\mathbf{M}^*$
```

Algorithm 6 **generateRule(T, P)**

```
Input: $T$: Rule templates; $P$ Set of open and closed predicates
Output: $r$: a rule
  $t \sim Unif(T)$
  Sample $p \subset P$ such that the predicates can be applied to t
  $r \leftarrow p$ applied to template $t$
  return $r$
```

Prior Template: For targets where we have no information, we typically have a prior value. This is captured by the prior template and has the following form:

$$\mathbf{P}_T(A, B) = \{0, 1\}$$

By setting different weights to these rules, we can vary the prior value for a targets in the range $[0, 1]$.

Having defined the set of rule templates, we introduce the *ESMS* algorithm which performs a structured search through the space of models in Algorithm 5. For each rule in the model, we first sample a template. We then sample predicates for each slot in the template. Once all the rules in the model are sampled, we learn the relative importance of these rules by performing weight learning. We use an efficient objective function, *Piecewise pseudolikelihood*, which is presented in the next subsection. We then evaluate the performance of the model $V(\mathbf{M})$ on the training data. We repeat this process N times and return the best performing model as the final model.

6.4.1 Piecewise pseudolikelihood

The *ESMS* algorithm involves learning the weights of the sampled rules. One approach to weight learning involves optimizing the likelihood function. However, the partition function Z in likelihood involves an integration that makes it intractable to compute. To overcome the intractable likelihood score, pseudo-likelihood [19] is commonly used in SRL structure learning and weight learning methods. For HL-MRFs, the pseudo-likelihood $\hat{P}_{\mathbf{M}}$ approximates the likelihood as:

$$\begin{aligned}
&\hat{P}_{\mathbf{M}}(\mathbf{Y}|\mathbf{X}) = \prod_{Y_i \in \mathbf{Y}} \frac{1}{Z_i(\mathbf{Y},\mathbf{X})} \exp(-E_i(Y_i,\mathbf{Y},\mathbf{X})) \\
&\text{where } Z_i(\mathbf{Y},\mathbf{X}) = \int_{Y_i} \exp(-E_i(Y_i,\mathbf{Y},\mathbf{X})) \qquad (6.4) \\
&E_i(Y_i,\mathbf{Y},\mathbf{X}) = \sum_{c \in C} \sum_{j : Y_i \in G_c} w_j \phi_j(Y_i,\mathbf{X},\mathbf{Y})
\end{aligned}$$

The notation $j : Y_i \in G_c$ selects ground clauses j where Y_i appears. However, due to the coupling of the rules, we also need to re-estimate the weights for the same rule in different models. Further, the objective function is non-convex and is hard to optimize.

To overcome these challenges, we propose a new, efficient-to-optimize objective function called **piecewise pseudolikelihood** (PPLL). PPLL has two key properties that makes weight learning highly scalable : 1) with PPLL, the optimal weight of a rule is independent of other rules in the model; and 2) the PPLL objective is convex and admits an inherently parallelizable gradient-based algorithm for optimization.

PPLL was first proposed for weight learning in conditional random fields (CRF) [231]. For HL-MRFs, PPLL factorizes the joint conditional distribution along both random variables and rules and is defined as:

$$\begin{aligned}
&\hat{P}_{\mathbf{M}}(\mathbf{Y}|\mathbf{X}) = \prod_{r \in R} \prod_{Y_i \in \mathbf{Y}} \frac{\exp(-E_i^r(Y_i, \mathbf{Y}, \mathbf{X}))}{Z_i^r(\mathbf{Y}, \mathbf{X})} \\
&\text{where } Z_i^r(\mathbf{Y}, \mathbf{X}) = \int_{Y_i} \exp(-E_i^r(Y_i, \mathbf{Y}, \mathbf{X})) \\
&E_i^r(Y_i, \mathbf{Y}, \mathbf{X}) = \sum_{j: Y_i \in G_r} w_j \phi_j(Y_i, \mathbf{Y}, \mathbf{X})
\end{aligned} \tag{6.5}$$

The key advantage of PPLL over likelihood arises from the factorization of Z into Z_i^c, which requires only ground rules corresponding to r and variable Y_i for its computation. Following standard convention, we optimize the log of PPLL denoted $l_{ppll}(\mathbf{w})$.

We now show that for the log PPLL objective function, performing weight learning on the entire model containing all rules is equivalent to optimizing the weight for each rule independently.

Theorem 3. *Optimizing $l_{ppll}(\boldsymbol{w})$ over the set of weights $\boldsymbol{w}$ is equivalent to optimizing over each w_r separately.*

Proof. By the definition of $l_{ppll}(\mathbf{w})$, we have

$$\begin{aligned}
\arg\max_{\mathbf{w}\in\mathbb{R}^+} l_{ppll}(\mathbf{w}) &= \arg\max_{\mathbf{w}\in\mathbb{R}^+} \sum_{r\in R}\sum_{Y_i\in\mathbf{Y}} log\frac{\exp(-E_i^r(Y_i,\mathbf{Y},\mathbf{X}))}{Z_i^r(\mathbf{Y},\mathbf{X})} \\
&= \sum_{r\in R} \arg\max_{w_r\in\mathbb{R}^+} \sum_{Y_i\in\mathbf{Y}} log\frac{\exp(-E_i^r(Y_i,\mathbf{Y},\mathbf{X}))}{Z_i^r(\mathbf{Y},\mathbf{X})} \\
&= \arg\max_{w_r\in\mathbb{R}^+} \sum_{Y_i\in\mathbf{Y}} log\frac{\exp(-E_i^r(Y_i,\mathbf{Y},\mathbf{X}))}{Z_i^r(\mathbf{Y},\mathbf{X})} \quad \forall r\in R
\end{aligned}$$

□

We optimize $l_{ppll}(\mathbf{w})$ using a projected gradient descent algorithm. The partial derivative of $l_{ppll}(\mathbf{w})$ for a given rule weight w_r is of the form:

$$\begin{aligned}
&\nabla_{w_r} = \Phi_r(Y_i,\mathbf{Y},\mathbf{X}) - \mathbb{E}_{ppll}[\Phi_r(Y_i,\mathbf{Y},\mathbf{X})] \\
&\text{where} \;\; \Phi_r(Y_i,\mathbf{Y},\mathbf{X}) = \sum_{Y_i\in\mathbf{Y}}\sum_{j:Y_i\in G_r} \phi_j(Y_i,\mathbf{Y},\mathbf{X})
\end{aligned} \tag{6.6}$$

The gradient for a rule weight w_r is the difference between observed and expected hinge-loss potential summed over corresponding ground rules G_r. We can compute observed penalties once and cache their values. Unlike the gradients for likelihood, each expectation term in the PPLL gradient considers a single rule. Thus, when evaluating gradients for weight updates, we use multi-threading to compute the expectation terms in parallel. The dual advantages of parallelizing and requiring weight learning only once for a rule makes PPLL highly scalable.

6.5 Explainabilty

Having learned a PSL model using the *ESMS* approach, we now describe our approach to generate explanations for the inferred values. The unobserved values are inferred by maximizing the likelihood of the graphical model. The value of a random variable Y_i depends on the all the hinge-loss clique potentials it is present in. We denote the set of all potentials Y_i participates in by Φ_{Y_i}. These potentials are generated by grounded templates that are interpretable. We can either display these ground templates directly to the user or use a translation system, that takes as input a ground template and outputs sentences in natural language or pictorially as described in Kouki et al. [134]. Thus the space of explanations $\mathcal{G}$ is given by Φ_{Y_i}.

6.5.1 End-User Explainability and Accuracy Trade-off

Certain data sources are easier to interpret than others by the end-user. For example, in a recommender system, rules containing predicates such as $SimilarUser_{Cosine}$ can be explained using sentences such as "*User U_1 who is similar to you liked this item I*". Other predicates such as latent factor recommendation approaches are hard to explain to the end-user. We partition the predicates into explainable and non-explainable predicates based on domain knowledge. We formally define end-user explainability of a rule as:

Definition 12 (α-explainable). *A rule r is α-explainable if the proportion of explainable predicates in the body of the rule is greater than α.*

Therefore, if a rule has no end-user explainable predicates in the body then it is a non-explainable (0-explainable) rule and if every predicate in the body of a rule is end-user explainable then it is a fully explainable (1-explainable) rule.

In applications where providing meaningful explanations to the end user is important, we may prefer models with high α-explainable. A model with high α-explainable rules can result in a greater number of predictions that are explainable. However, this might result in a loss of predictive accuracy. To address this trade-off at the model discovery time, we introduce an explainability parameter $\gamma \in [0, 1]$

which is the minimum proportion of rules in a model that are explainable and tune it based on the application's need. In the while loop of Algorithm 5, with probability γ, we generate an α-explainable by repeatedly generating rules until we meet the explainability criteria; with probability $1-\gamma$, we simply add the next generated rule to the model. A value of 1 for γ ensures that every rule in the model only contains predicates that are explainable and hence all predictions can be explained. This ensures that the generated explanations satisfy the property of *explicitness*.

The set of ground rules that correspond to the space of explanations is usually very large and not all are equally important. To ensure *faithfulness*, we measure the importance of each ground rule to the inferred value (using the explaining function f), and display the most important rule to the user.

Definition 13. *The explaining function $f_i : (\mathbf{X}, Y, \Phi_{Y_i}) \to \mathbb{R}$ scores the importance of a ground rule $\phi_j \in \Phi_{Y_i}$ with respect to a random variable Y_i. It is given by the norm of the first partial derivative of the ground rule at the inferred value y_i, i.e:*

$$f_i(\mathbf{X}, \mathbf{Y}, \phi_j) = \left\| \frac{w_j \partial \phi_j(\mathbf{X}, \mathbf{Y})}{\partial Y_i} |_{y_i} \right\|_1 \tag{6.7}$$

Modifying a rule with high f_i would lead to a significant change in the RV's inferred value.

6.5.2 Stability of the explanation function

Having shown that explaining function f satisfies the first two properties, we now show that it is stable as defined in Section 6.3.

We first review the definition of strong convexity and observe that the PSL energy function is strongly convex.

Definition 14. *A function $E : (\mathcal{Y}, \mathcal{X}) \rightarrow \mathbb{R}$ is κ-strongly convex in $\mathcal{Y}$ (w.r.t the 1-norm) if $\mathcal{Y}$ is a convex set and, for any $\mathbf{Y}, \mathbf{Y}' \in \mathcal{Y}$, $\tau \in [0, 1]$,*

$$\tau(1-\tau)\frac{\kappa}{2}\|\mathbf{Y}-\mathbf{Y}'\| + E(\tau\mathbf{Y} + (1-\tau)\mathbf{Y}', \mathbf{X}) \leq \tau E(\mathbf{Y}, \mathbf{X}) + (1-\tau)E(\mathbf{Y}', \mathbf{X})$$

for $\mathbf{X} \in \mathcal{X}$.

The energy function E is a summation of squared hinges and hence E is convex. Further, the prior template described in Section 6.4 acts a regularizer of $\mathbf{Y}$ and is κ-strongly convex. Hence E is at least κ-strongly convex in $\mathcal{Y}$ [154].

We next state and prove two lemmas that show the change in the optimal energy function is bounded when the value of one of the observed RV ($\mathbf{X}$) is changed (Lemma 1) and this bounds the change is the unobserved RVs $\mathbf{Y}$s (Lemma 2).

Lemma 1. *For a graphical model G with a set of potentials $\mathbf{\Phi}$, let Q_i denote the number of potentials that involve $\mathbf{X_i}$, and let $Q_G \triangleq \max_i Q_i$. Let $\|\mathbf{w}\| < R$. Let $\mathbf{X}, \mathbf{X}' \in \mathcal{X}$ differ at a single coordinate i by atmost ϵ. Then, for $\dot{\mathbf{Y}} \triangleq \arg\min_{\mathbf{Y}} E(\mathbf{Y}, \mathbf{X})$ and $\dot{\mathbf{Y}}' \triangleq \arg\min_{\mathbf{Y}} E(\mathbf{Y}, \mathbf{X}')$,*

$$\left\|E(\dot{\mathbf{Y}}', \mathbf{X}) - E(\dot{\mathbf{Y}}', \mathbf{X}')\right\| \leq \epsilon R\sqrt{Q_G}$$

Proof.

$$\begin{aligned}
&\left\|E(\dot{\mathbf{Y}}', \mathbf{X}) - E(\dot{\mathbf{Y}}', \mathbf{X}')\right\| \\
&= \left\|\mathbf{w}^T \mathbf{\Phi}(\dot{\mathbf{Y}}', \mathbf{X}) - \mathbf{w}^T \mathbf{\Phi}(\dot{\mathbf{Y}}', \mathbf{X}')\right\| \\
&\leq \|\mathbf{w}\| \left\|\mathbf{\Phi}(\dot{\mathbf{Y}}', \mathbf{X}) - \mathbf{\Phi}(\dot{\mathbf{Y}}', \mathbf{X}')\right\| \text{ [Form Cauchy-Schwarz]} \\
&\leq R \left\|\mathbf{\Phi}(\dot{\mathbf{Y}}', \mathbf{X}) - \mathbf{\Phi}(\dot{\mathbf{Y}}', \mathbf{X}')\right\|
\end{aligned}$$

because, by definition, $\|\mathbf{w}\|$ is upper bounded by R. Note that $\mathbf{\Phi}(\dot{\mathbf{Y}}', \mathbf{X})$ and $\mathbf{\Phi}(\dot{\mathbf{Y}}', \mathbf{X}')$ only differ at any grounding involving $\mathbf{X_i}$. The number of such groundings is Q_i, which is upper-bounded by Q_G, so at most Q_G potentials will change.

Further, the squared hinge loss potential has the from $max\{\mathbf{C_x}^T\mathbf{X} + \mathbf{C_y}^T\mathbf{Y} - c, 0\}^2$ where $\mathbf{C_x}, \mathbf{C_y}$ co-efficient vectors consisting of $1, -1, 0$.

$$\begin{aligned}
&\left\|\mathbf{\Phi}(\dot{\mathbf{Y}}', \mathbf{X}) - \mathbf{\Phi}(\dot{\mathbf{Y}}', \mathbf{X}')\right\| \\
&= \left(\sum_{\phi \in \mathbf{\Phi}} \mathbb{1}\{\mathbf{C_{X_i}} \neq 0\}((\phi(\dot{\mathbf{Y}}', \mathbf{X}) - \phi(\dot{\mathbf{Y}}', \mathbf{X}')))^2\right)^{1/2} \\
&= \left(\sum_{\phi \in \mathbf{\Phi}} \mathbb{1}\{\mathbf{C_{X_i}} \neq 0\}(max\{\mathbf{C_x}^T\mathbf{X} + \mathbf{C_y}^T\dot{\mathbf{Y}}' - c, 0\} - max\{\mathbf{C_x}^T\mathbf{X}' + \mathbf{C_y}^T\dot{\mathbf{Y}}' - c, 0\})^2\right)^{1/2} \\
&\leq \left(\sum_{\phi \in \mathbf{\Phi}} \mathbb{1}\{\mathbf{C_{X_i}} \neq 0\}(max\{\mathbf{C_x}^T(\mathbf{X} - \mathbf{X}') + \mathbf{C_y}^T(\dot{\mathbf{Y}}' - \dot{\mathbf{Y}}'), 0\})^2\right)^{1/2} \\
&\leq (Q_i)^{1/2}\,\epsilon \leq (Q_G)^{1/2}\,\epsilon
\end{aligned}$$

□

Lemma 2. *Let $E : (\mathcal{Y}, \mathcal{X}) \to \mathbb{R}$ be κ-strongly convex, and let $\dot{\mathbf{Y}} \triangleq \arg\min_{\mathbf{Y}} E(\mathbf{Y}, \mathbf{X})$ and $\dot{\mathbf{Y}}' \triangleq \arg\min_{\mathbf{Y}} E(\mathbf{Y}, \mathbf{X}')$, where $\mathbf{X}, \mathbf{X}' \in \mathcal{X}$ differ at a single coordinate i.*

Then,

$$\left\|\dot{\mathbf{Y}}' - \dot{\mathbf{Y}}\right\|_1^2 \leq \frac{2}{\kappa}|E(\dot{\mathbf{Y}}', \mathbf{X}) - E(\dot{\mathbf{Y}}', \mathbf{X}')| \tag{6.8}$$

Proof. Without loss of generality, assume that $E(\dot{\mathbf{Y}}, \mathbf{X}) \geq E(\dot{\mathbf{Y}}', \mathbf{X}')$.(If $E(\dot{\mathbf{Y}}, \mathbf{X}) \leq E(\dot{\mathbf{Y}}', \mathbf{X}')$ we can state this in terms of $\dot{\mathbf{Y}}'$). Let $\Delta\mathbf{Y} \triangleq \dot{\mathbf{Y}}' - \dot{\mathbf{Y}}$. By Definition 14, for any $\tau \in [0, 1]$,

$$\tau(1-\tau)\frac{\kappa}{2}\left\|\dot{\mathbf{Y}}' - \dot{\mathbf{Y}}\right\| + E(\tau\dot{\mathbf{Y}}' + (1-\tau)\dot{\mathbf{Y}}, \mathbf{X}) \leq \tau E(\dot{\mathbf{Y}}', \mathbf{X}) + (1-\tau)E(\dot{\mathbf{Y}}, \mathbf{X})$$

Since $\dot{\mathbf{Y}}$ is, by definition, the unique minimizer of $E(\mathbf{Y}, \mathbf{X})$, it follows that $E(\dot{\mathbf{Y}} + \tau\Delta\mathbf{Y}, \mathbf{X}) - E(\dot{\mathbf{Y}}, \mathbf{X}) \geq 0$, so the above inequality is preserved when this term is dropped. This, dividing both sides by $\tau\kappa/2$, we have that

$$\begin{aligned}(1-\tau)\left\|\Delta\mathbf{Y}\right\|^2 &\leq \frac{2}{\kappa}(E(\dot{\mathbf{Y}}', \mathbf{X}) - E(\dot{\mathbf{Y}}, \mathbf{X})) \\ \left\|\Delta\mathbf{Y}\right\|^2 &\leq \frac{2}{\kappa}(E(\dot{\mathbf{Y}}', \mathbf{X}) - E(\dot{\mathbf{Y}}, \mathbf{X}))\end{aligned}$$

where the last inequality follows from the fact that $(1-\tau)$ is maximized at $\tau = 0$. Since $E(\dot{\mathbf{Y}}, \mathbf{X}) \geq E(\dot{\mathbf{Y}}', \mathbf{X}')$, the following inequality holds

$$\left\|\dot{\mathbf{Y}}' - \dot{\mathbf{Y}}\right\|^2 \leq \frac{2}{\kappa}(E(\dot{\mathbf{Y}}', \mathbf{X}) - E(\dot{\mathbf{Y}}', \mathbf{X}'))$$

□

We now state a lemma that shows that the change in the explaining function score for a ground rule j with respect to the observed RV Y_k denoted by $f_k(\mathbf{X}, \mathbf{Y}, \phi_j)$ is bounded.

Lemma 3. *Let the explaining function f be defined as $f(\mathbf{X}, \mathbf{Y}, \phi) = \left\| \frac{w\partial\phi(\mathbf{X},\mathbf{Y})}{\partial \mathbf{Y_i}}|_y \right\|$. Let $\mathbf{X}, \mathbf{X'} \in \mathcal{X}$ differ at a single random variable $\mathbf{X_i}$ by at most ϵ. Let $\|\mathbf{Y} - \mathbf{Y'}\| < B$ for any two $\mathbf{Y}, \mathbf{Y'} \in \mathcal{Y}$ and $\|\mathbf{w}\| < R$. Then:*

$$|f(\mathbf{X}, \mathbf{Y}, \phi) - f_k(\mathbf{X'}, \mathbf{Y'}, \phi)| \leq 2R(\epsilon + B) \tag{6.9}$$

Proof. The hinge loss function ϕ has the from $max\{\mathbf{C_x}^T\mathbf{X} + \mathbf{C_y}^T\mathbf{Y} - c, 0\}^2$ where $\mathbf{C_x}, \mathbf{C_y}$ co-efficient vectors consisting of $1, 0, -1$.

The partial derivative w.r.t to $\mathbf{Y_i}$

$$\begin{aligned}
&= \left\| \frac{\partial\phi(\mathbf{X}, \mathbf{Y})}{\partial \mathbf{Y_i}}|_y \right\| \\
&= 2 * max\{\mathbf{C_x}^T\mathbf{X} + \mathbf{C_y}^T\mathbf{Y} - c, 0\} * \left\| \frac{\partial max\{\mathbf{C_x}^T\mathbf{X} + \mathbf{C_y}^T\mathbf{Y} - c, 0\}}{\partial \mathbf{Y_i}}|_y \right\| \\
&= 2 * max\{\mathbf{C_x}^T\mathbf{X} + \mathbf{C_y}^T\mathbf{Y} - c, 0\}
\end{aligned}$$

Now consider $|f(\mathbf{X}, \mathbf{Y}, \phi) - f(\mathbf{X'}, \mathbf{Y'}, \phi)|$

$$\begin{aligned}
&= w \left\| \frac{\partial\phi(\mathbf{X}, \mathbf{Y})}{\partial \mathbf{Y_i}}|_y - \frac{\partial\phi_j(\mathbf{X'}, \mathbf{Y'})}{\partial \mathbf{Y_i}}|_{y'} \right\| \\
&= 2w\|max\{\mathbf{C_x}^T\mathbf{X} + \mathbf{C_y}^T\mathbf{Y} - c, 0\} - max\{\mathbf{C_x}^T\mathbf{X'} + \mathbf{C_y}^T\mathbf{Y'} - c, 0\}\| \\
&\leq 2w \left\| max\{\mathbf{C_x}^T(\mathbf{X} - \mathbf{X'}) + \mathbf{C_y}^T(\mathbf{Y} - \mathbf{Y'}), 0\} \right\| \\
&\leq 2w \|max\{\epsilon + B, 0\}\| \\
&\leq 2R(\epsilon + B)
\end{aligned}$$

□

We now prove that scores by the explaining function f is stable.

Theorem 4. *The explaining function f_i is stable with respect to $h(\mathbf{X}, \mathbf{Y})$.*

	Cora	Yelp		LastFM	
	AUPR	MAE	MSE	MAE	MSE
BOOST	**0.700** (0.163)	**0.196** (0.008)	0.079 (0.007)	0.279 (0.058)	0.11 (0.044)
BOOST$_{PPLL}$	**0.651** (0.186)	0.212 (0.012)	0.092 (0.013)	0.257 (0.046)	0.0941 (0.058)
PRA	0.622 (0.169)	0.2005 (0.0004)	0.086 (0.0004)	0.186 (0.001)	0.048 (0.0004)
ESMS	**0.684** (0.148)	**0.193** (0.003)	**0.065** (0.008)	**0.177** (0.070)	**0.043** (0.0004)

Table 6.1: Metrics: Our *ESMS* approach significantly outperforms other approaches on recommendation datasets and is comparable to BOOST on Cora. Numbers in bold are statistically significant with $p < 0.05$.

Proof. From Lemma 1 and Lemma 2, for any $\mathbf{X}, \mathbf{X}' \in \mathcal{X}$ that differ in a single random variable i, we have

$$|E(\dot{\mathbf{Y}}', \mathbf{X}) - E(\dot{\mathbf{Y}}', \mathbf{X}')| \leq RQ_G$$
$$\Rightarrow \left\|\dot{\mathbf{Y}}' - \dot{\mathbf{Y}}\right\|_1 \leq \sqrt{\frac{2}{\kappa} RQ_G}$$

From Lemma 3, we have

$$|f_k(\mathbf{X}, \dot{\mathbf{Y}}, \phi_j) - f_k(\mathbf{X}', \dot{\mathbf{Y}}', \phi_j)| \leq 2R(1 + \sqrt{\frac{2}{\kappa} RQ_G})$$
$$= |f_k(\mathbf{X}, h(\mathbf{X}), \phi_j) - f_k(\mathbf{X}', h(\mathbf{X}'), \phi_j)| \leq D$$

Thus f is stable with respect to $E(\mathbf{Y}, \mathbf{X})$. □

6.6 Experimental Evaluation

We investigate the following research questions empirically:

- RQ1) What is the predictive accuracy of models discovered by *ESMS* ?
- RQ2) What is the impact of the explainability parameter γ on end-user explainability?

Entity Resolution Predicates	
$SAME\_AUTHOR$	$SAME\_BIB$
$SAME\_VENUE$	$SAME\_TITLE$
$AUTHOR$	$VENUE$
$TITLE$	$HASWORD\_AUTHOR$
$HASWORD\_TITLE$	$HASWORD\_VENUE.$

Table 6.2: Entity resolution predicates: List of predicated for the entity resolution datasets.

- RQ3) How well can the predictions be explained?

6.6.1 Data

We evaluate the predictive accuracy of the discovered models on an entity resolution dataset and two recommendation datasets. Further, for the recommendation datasets, we evaluate the generated explanations.

Enity Resolution Dataset: The Cora entity resolution dataset is based on the citation references between scientific papers. The task is to identify papers that are the same. This is represented by the target predicate $SAME\_BIB$. The dataset contains 10 predicates and are shown in Table 6.2. For each of the non-target predicates, we included the inverse predicates where the arguments are reversed. For example, for the predicate $SAME\_AUTHOR(A, B)$ we include the predicate $\_SAME\_AUTHOR(B, A)$. In total there are 19 predicates. Since all predicates are explainable, we do not classify them as explainable and non-explainable predicates. The dataset is split into 5 folds. We use the same splits as Khot et al. [120].

Recommendation Dataset: We run experiments on two recommendation datasets **Yelp** and **LastFM** . **Yelp** is a restaurant recommendation dataset containing 34,454 users, 3,605 restaurants, 8,512 friendship links and 99,049 observed ratings. **LastFM** is a music artist recommendation dataset containing 1,892

Explainable Predicates	
USERS_ARE_FRIENDS	*SIM_COSINE_ITEMS*
SIM_PEARSON_ITEMS	*SIM_CONTENT_ITEMS_JACCARD*
SIM_ADJCOS_ITEMS	*SIM_MF_COSINE_ITEMS*
SIM_MF_EUCLIDEAN_ITEMS	*SIM_COSINE_USERS*
SIM_PEARSON_USERS	*SIM_MF_COSINE_USERS*
SIM_MF_EUCLIDEAN_USERS	*AVG_ITEM_RATING*
RATING_PRIOR	*AVG_USER_RATING*

Table 6.3: Expainable predicates: List of explainable predicated for the recommendation datasets.

Non-explainable Predicates	
RATING	*RATED*
SGD_RATING	*BPMF_RATING*
ITEM_PEARSON_RATING	*USER*
ITEM	

Table 6.4: Non-expainable predicates: List of non-explainable predicated for the recommendation datasets.

users, 17,632 music artists, 12,717 friendship links and 92,834 observed ratings.

Both **Yelp** and **LastFM** datasets contain 21 predicates or relations. We categorize the predicates as explainable and non-explainable predicates based on how easy it is for an end-user to understand the predicates. There were 14 explainable predicates and 7 non-explainable predicates. The list of explainable predicates are shown in Table 6.3 and the list non-explainable predicates is Table 6.4.

The **Yelp** dataset is split in five folds. Each fold contains a train and a test split. The train splits contains 79240 observed ratings and 7924 ratings that need to be predicted. The test split contains 99049 observed ratings and 19809 ratings that need to be predicted.

Similarly, the **LastFM** dataset is split in five folds. Each fold contains a train and a test split. The train splits contains 74267 observed ratings and 18567 ratings that need to be predicted. The test split contains 92834 observed ratings and 18567 ratings that need to be predicted.

For both datasets, the task is to predict the unobserved ratings. To prevent the generation of a quadratic number of user-item pairs, we perform *blocking.* Blocking restricts the rating pairs by identifying the *important pairs* using a simple heuristic. For details see Augustine and Getoor [12]. We use the splits from Kouki et al. [133].

6.6.2 Approaches

We evaluate by comparing the following structure learning methods:

BOOST[120]: This is a state-of-the-art structure learning approach for MLNs. It uses Friedman's functional gradient boosting algorithm to generate a series of relational regression problems, which in turn are used to generate the rules in the model. We use the code of [120] with the recursion flag set to True.[1]. BOOSTuses Boolean logic, so we round the values of the ground atoms to 1 if the value is greater that 0.5, and 0 otherwise. We learn 10 trees and combined the rules across the trees to generate a PSL model. We use the same weights learned by the BOOST approach. Since PSL only allows positive weights, we truncate negative weights to 0. In addition, we also evaluate a model with the weights learned using the PPLL objective (BOOST$_{PPLL}$).

PRA[85]: PRA is a relational path finding algorithm that identifies paths that connect unobserved pairs by performing random walks. We use the code of Gardner and Mitchell [85] to identify paths of length up to three [2]. We then convert these paths to PSL rules. We learn the rule weights using our proposed PPLL weight learning method.

ESMS: Our proposed approach that performs a structured search to learn an explainable PSL model. We use the rule templates described in 6.4. We set the maximum number of rules in a model to 15, maximum iterations to 100 and $\gamma = 0$.

[1]https://starling.utdallas.edu/software/boostsrl

[2]https://matt-gardner.github.io/pra/

6.6.3 Predictive performance of *ESMS*

We evaluate [RQ1] by comparing the predictive accuracies of BOOST, BOOST$_{PPLL}$, PRA and *ESMS* . We compute the postive class AUPR for the CORA dataset. For the recommendation datasets we compute the mean squared error (MSE) and mean absolute error (MAE) by rescaling the ratings between $[0, 1]$. [133]. Table 6.1 shows the performance metrics computed across the 5 folds. We report the mean and standard deviation. We perform a paired t-test to measure significance and the numbers in bold are statistically significant with $p < 0.05$. First, we observe that the *ESMS* approach outperforms both versions of BOOST and PRA on the recommendation datasets. On the entity resolution dataset, it outperforms PRA and is comparable to BOOST. PRA can only discover rules that are paths and this limitation hurts the performance of the model. We next observe that the BOOST models perform better than the BOOST$_{PPLL}$ model. The BOOST method did not learn any collective rules such as: $Rating(A, B) \wedge SimItem(B, C) \rightarrow Rating(A, C)$. These content-based rules are important for the recommender system performance. As a result, the *ESMS* performs better than BOOST. The learned models for all approaches are given in the supplementary material.

ESMS discovered social rules such as $Friends(U_1, U_2) \wedge Rating(U_1, I_1) \rightarrow Rating(U_2, I_1)$, similarity rules such $SimItem_{Pearson}(I_1, I_2) \wedge Rating(U_1, I_1) \rightarrow Rating(U_1, I_2)$. Further, the model incorporates external systems with rules such as $BPMF(U_1, I_1) \rightarrow Rating(U_1, I_1)$.

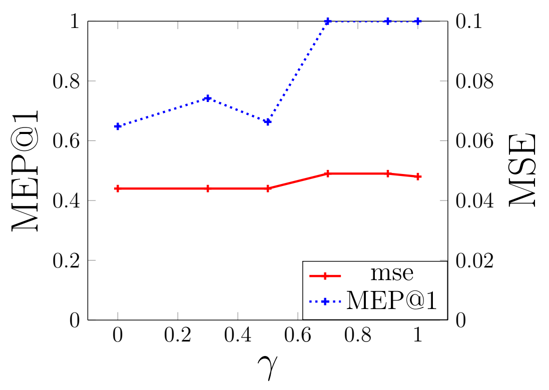


Figure 6.1: MEP vs MSE for LastFM: As we increase γ we generate more explainable models that have a slightly higher MSE

6.6.4 Trade-off between predictive accuracy and explainability

We evaluate [RQ2] by investigating the impact of the explainability parameter γ on a model's predictive accuracy and end-user explainability. For each prediction, we generated a ranked list of ground rules in $\mathbf{G_i}$ and compute *mean explainable precision*(MEP@K) [2] that represents that fraction of ratings that are explainable. MEP@K is defined as $\frac{1}{|\mathbf{Y}|}\sum_{i=1}^{|\mathbf{Y}|}(\mathcal{E}_k(\mathbf{G_i}))$, where $\mathcal{E}_k(\mathbf{G_i})$ is one if one of the top-k ranked rules in $\mathbf{G_i}$ is explainable and zero otherwise. We assume a rating to be explainable if it contains at least one explainable predicate ($\alpha = 0.25$). We modified γ from 0 to 1 ($\gamma = \{0, 0.3, 0.5, 0.7, 0.9, 1.0\}$) and computed the MEP@1 and MSE of all generated models.

Fig. 6.1 shows the change in MSE and MEP@1 as we vary γ for the LASTFM dataset. We observe that, not surprisingly, we generate models with more explainable rules as we increase γ. However, the MSE also increases slightly. This due to the model not containing non-explainable rules such as latent factor models that have high predictive accuracy. We found a similar pattern on the YELP dataset.

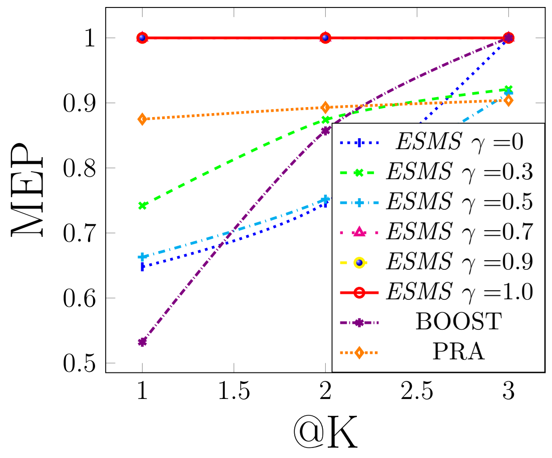


Figure 6.2: MEP for LastFM: MEP increases for all approaches as we increase K. *ESMS* with $\gamma > 0.7$ outperforms BOOSTand PRA.

Analysis of Explanations

We evaluate [RQ3] by analyzing the MEP for all models at $K = \{1, 2, 3\}$. Fig. 6.2 shows the MEP@K for various approaches. As we increase the value for K, the MEP value increases for all approaches. For *ESMS* , we get a MEP of 1 for $\gamma > 0.7$ for all K. PRA has MEP close to 0.9 due to the large number of rules in the model. BOOST starts with MEP close to 0.5 at $K = 1$ but increases rapidity as we increase K.

As a concrete example of our results, we look at an example of a ground rule that was identified by our approach as the most important explanation for a rating in the LastFM dataset. For pair $(User12, Artist5)$ *ESMS* identified the most important rule as: $MF(User12, Artist5) \rightarrow Rating(User12, Artist5)$ when γ was set to 0. This is a non-explainable rule. However, when we change $\gamma = 1$, the most important ground rule became: $Rating(User12, Artist29) \wedge SimItem_{jaccard}(Artist29, Artist5) \rightarrow Rating(User12, Artist5)$. This is explainable.

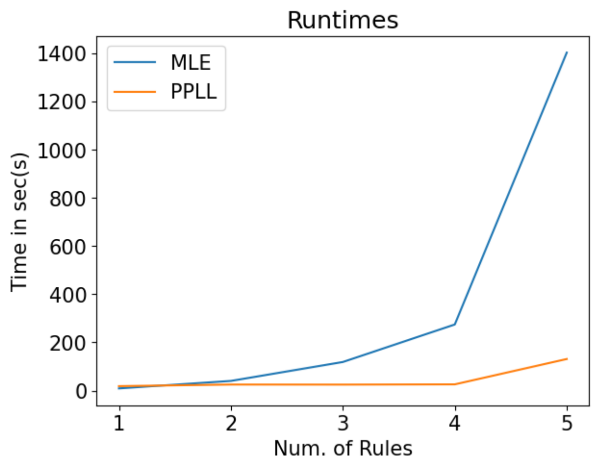


Figure 6.3: Running time for weight learning: Runtime increase exponentially for MLE but increases linearly for PPLL.

6.6.5 Timing Experiment

We evaluate the runtimes for the proposed PPLL weight learning approach and the standard Maximum Likelihood Estimate (MLE) approach. Given a set of rules, PPLL compute the weights only once for each rule. The presence of other rules in the model does not affect the weight of a rule. However, since MLE couples all the rules, we need to compute the weights for each subset of the rules and select the best model.

Fig. 6.3 shows the runtimes in seconds for PPLL and MLE as the number of rules in the model increases from 1 to 5. We observe that the runtimes increase exponentially for MLE but increases linearly for PPLL. The decoupling of the rules in weight learning help scale our approach to models with larger sets of rules.

6.7 Related Work

We give a brief overview of structure learning in templated graphical models and related work in explainability.

6.7.1 Structure Learning:

Many algorithms have been proposed to learn MLNs, a class of discrete TGMs. Bottom-up approaches generate informative clauses by using relational paths to capture patterns and motifs in the data [162, 123, 124]. Most recently, MLN structure learning has been viewed from the perspectives of moralizing learned Bayesian networks [119] and functional gradient boosting [120]. These methods improve scalability while maintaining predictive performance. Structure learning methods for specific to a task of interest use inductive logic programming [169] to generate clauses which are pruned with L1-regularized learning [111, 112] or perform iterative local search [22] to refine rules with the operations described above. For PSL, a reinforcement learning based approach has been proposed [266]. Our approach builds on these approaches and the rule templates [207, 249, 251] to learn an model that also generates explanations.

6.7.2 Explainability:

Explainable models can be broadly classified into model-intrinsic methods and model-agnostic methods. Model-intrinsic approaches such as Catherine and Cohen [37], Kouki et al. [134], Al-Shedivat et al. [8] use interpretable models that are easy to explain. Model-agnostic or post-hoc explanations such as Ribeiro et al. [204], Peake and Wang [185], Yang et al. [259] consider the model as a black box and generate explanations from the output. Our proposed approach is a model-intrinsic method that learns an interpretable PSL model. Several gradient-based and perturbation based explanations have been proposed by Bach et al. [13], Zeiler and Fergus [263], Shrikumar et al. [222], Wolf et al. [252] for deep learning models. In this work, we propose a similar approach for templated graphical models.

6.8 Conclusion and Future Work

We proposed an efficient approach to learn explainable templated graphical models that trades-off between performance and explainability. Our explanation framework satisfies the properties of explicitness, faithfulness and stability. Our work suggests interesting future directions. Latent predicates are crucial for improving model performance. We plan to extend our approach to models with latent predicates. Our approach to rank explanations does not account for end-user preferences. These preferences could be incorporated into the ranking algorithms.

Chapter 7

Conclusion and Future Work

In this dissertation, I have proposed the three important knowledge discovery tasks of data alignment, estimating aggregate graph queries and model discovery for knowledge graphs and developed novel approaches for these tasks. The tasks capture both the local properties of the KG such as relationships that exist between two entities of same type and global graph properties such as logical rules and subgraph patterns that hold across several entities and relations of different types. These approaches address many of the practical challenges such as the presence of rich sets of entities and relations and lack of training data and negative instances. I have evaluated both statistical relational learning and deep learning based approaches' ability to address these challenges. Further, I compare and contrast these approaches both empirically and theoretical. In next two section, I briefly summarize the my contributions and propose future research directions to expand this thesis.

7.1 Summary of Contributions

In the area of data alignment, I first proposed an approach to align entities with duplicate and variation relationships. My contributions include: (1) extending the traditional task of two class entity linkage, where the goal is to identify duplicates, to a three class setting, where along with duplicates we also identify entity variations. (2) proposing a variational attribute discovery approach, **VarSpot**, which is scalable and unsupervised. The approach analyzes similarities and differences within the *same* catalog. (3) I proposed the notion of *contrast features* to model variational attributes. (4) I performed empirical evaluation in three different domains to show the generality of the approach. Using three different state-of-the-art entity linkage frameworks, including rule-based and deep learning based frameworks, I showed that models with contrast features significantly outperform models without them when identifying duplicates and variations. and (5) I evaluated the interpretable nature of contrast features through Mechanical Turk annotation experiments.

I then extended the alignment relationships to more fine-grained attributes and proposed an approach to identify and extract the discriminative attribute the distinguishes these entities and the values for these attributes. My contributions in this area include: (1) I proposed a taxonomy of approaches by extending the state-of-the-art attribute identification and attribute value extraction approaches for the discriminative attribute extraction task. (2) I proposed a novel end-to-end multitask approach that jointly performs both the discriminative attribute identification and value extraction tasks. I establish a theoretical upper bound for a class of approaches called extraction-oriented approaches for the attribute identification task. (3) I introduced a novel dataset and empirically show that the proposed **DiffXtract** approach outperforms attribute-oriented and extraction-oriented approaches.

In the area of estimating global graph properties, I introduced the concept of aggregate graph queries and compared various approaches to estimate them when the knowledge graph is incomplete with missing node labels. My contributions in this area include: (1) I introduced a suite of practical aggregate graph queries that measures the key graph property of subgroup cohesion and study the effectiveness of statistical relational learning and graph neural networks in estimating them. (2) I showed that first inferring the missing values and then estimating the aggregate graph queries leads to poor performance. (3) I proposed a novel Metropolis-within-Gibbs sampling framework, *MIG*, for probabilistic soft logic that is faster than existing statistical relational learning samplers. (4) Through experiments on three benchmark datasets, I showed that computing aggregate properties as an expectation outperforms point estimate approaches.

Finally, in the area of model discovery, I proposed an approach to discovery explainable models that can infer missing relationships in the knowledge graph. Further, I showed that these models can also generate explanations for the inferred relationships. My contributions in this area include: (1) I proposed a novel structured search approach that efficiently discovers a templated graphical model using rule templates that best captures the statistical dependencies in the data. (2) I introduced an efficient weight learning strategy based on a piecewise pseudo-likelihood objective that allows parallelization and requires weights for a template to be learned only once across models. (3) Using an explainability parameter, the proposed learning approach generates models that trade-off accuracy and end-user explainability of its predictions. (4) I proposed a new Fisher score based ranking algorithm that identifies the best explanation for a prediction and theoretically show that this is stable. (5) Empirically I showed, that the discovered models using the proposed approach outperform models generated using other state-of-the-art methods.

7.2 Future Work

While this work addressed the core challenges in estimating the local and global properties of knowledge graphs, there are several future avenues to expand this work. Specifically, three areas of research that seems promising to me are:

Incorporating user preferences: Many of the approaches proposed in this dissertation make assumptions that are often subjective and should ideally incorporate user preferences. For example, when identifying variations the distinction between base attributes and variational attributes is often be subjective and varies across categories. For some consumers, the distinction between a low-sodium versus regular soup is irrelevant;for others it is highly important. Similarly brand is an important base attribute for electronics but might be a variational attribute for medicines. Similarly, for the task of generating explanations, the set of predicates that are explainable and those that are hard to understand by the end-user is user and domain specific. Incorporating these preferences into approaches can greatly expand the scope of these approaches.

Using knowledge graph embeddings: Knowledge graph embedding have emerged as a method of choice for incorporating nodes and relations in the graph into various information retrieval and machine learning tasks. Some of the proposed approaches in this dissertation, such as **DiffXtract**, use word embeddings to capture the semantic information in the node attributes such as title. However, there is scope for learning embeddings for entities and relations in the graph using all the attributes and the relationships, and then use these embedding for the tasks proposed in this dissertation.

Neuro-symbolic approaches: While some approaches proposed in this dissertation are symbolic, some of the proposed approaches make use of sub-symbolic neural approaches. One promising area that has been gaining importance recently is the neuro-symbolic approach that integrates the low-level representational power of deep neural networks with high-level symbolic reasoning. These approaches leverage the power of deep neural networks in learning intermediate representations and combine them with the ability of symbolic approaches to incorporate domain knowledge and common-sense reasoning. An interesting future direction is to explore the ability of these approaches to solve the tasks proposed in this dissertation. For example, while estimating aggregate graph queries, we could combine graph neural network based approaches with statistical relational learning models to incorporate node representations and also infer a joint distribution over the unobserved data.

Bibliography

[1] Emmanuel Abbe. Community detection and stochastic block models: Recent developments. *JMLR*, 18:1–86, 2018.

[2] Behnoush Abdollahi and Olfa Nasraoui. Explainable matrix factorization for collaborative filtering. In *WWW Companion*, 2016.

[3] Behnoush Abdollahi and Olfa Nasraoui. Using explainability for constrained matrix factorization. In *RecSys*, 2017.

[4] Bijaya Adhikari, Yao Zhang, Aditya Bharadwaj, and B Aditya Prakash. Condensing temporal networks using propagation. In *ICDM*, 2017.

[5] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, 17(6):734–749, 2005.

[6] Charu C Aggarwal. *Data mining: the textbook*. Springer, 2015.

[7] Charu C Aggarwal. Content-based recommender systems. In *Recommender Systems*, pages 139–166. Springer, 2016.

[8] Maruan Al-Shedivat, Avinava Dubey, and Eric Xing. Contextual explanation networks. *Journal of Machine Learning Research*, 21(194):1–44, 2020.

[9] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. In *NeuRIPs*, 2018.

[10] Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, 2002.

[11] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[12] Eriq Augustine and Lise Getoor. A Comparison of Bottom-Up Approaches to Grounding for Templated Markov Random Fields. In *SysML*, 2018.

[13] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS*, 10(7), 2015.

[14] Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *JMLR*, 18:1–67, 2017.

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[16] Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews. In *KDD*, 2017.

[17] François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2rdf: towards a mashup to build bioinformatics knowledge systems. *Journal of biomedical informatics*, 41(5):706–716, 2008.

[18] Islam Beltagy, Katrin Erk, and Raymond Mooney. Probabilistic soft logic for semantic textual similarity. In *ACL*, 2014.

[19] Julian Besag. Statistical analysis of non-lattice data. *JRSS*, 24(3):179–195, 1975.

[20] Indrajit Bhattacharya and Lise Getoor. Iterative record linkage for cleaning and integration. In *SIGMOD workshop on DMKD*, 2004.

[21] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. In *TKDD*, 2007.

[22] Marenglen Biba, Stefano Ferilli, and Floriana Esposito. Discriminative structure learning of Markov logic networks. In *ILP*, 2008.

[23] P Bickel, P Diggle, S Fienberg, U Gather, I Olkin, and S Zeger. *Springer Series in Statistics*. Springer, 2009.

[24] Mikhail Bilenko, Beena Kamath, and Raymond J Mooney. Adaptive blocking: Learning to scale up record linkage. In *ICDM*, 2006.

[25] Mustafa Bilgic and Raymond J Mooney. Explaining recommendations: Satisfaction vs. promotion. In *IUI Workshop on Beyond Personalization*, 2005.

[26] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[27] Béla Bollobás, Svante Janson, and Oliver Riordan. The phase transition in inhomogeneous random graphs. *RSA*, 31:3–122, 2007.

[28] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NeuRIPS*, 2013.

[29] Svetlin Bostandjiev, John O'Donovan, and Tobias Höllerer. Tasteweights: a visual interactive hybrid recommender system. In *RecSys*, 2012.

[30] David Guy Brizan and Abdullah Uz Tansel. A. survey of entity resolution and record linkage methodologies. *Communications of the IIMA*, 6(3):5, 2006.

[31] Matthias Broecheler and Lise Getoor. Computing marginal distributions over continuous markov networks for statistical relational learning. In *NeuRIPS*, 2010.

[32] T N Bui, S Chaudhuri, F T Leighton, and M Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7:171–191, 1987.

[33] Robin Burke. Knowledge-based recommender systems. *ELIS*, 69 (Supplement 32):175–186, 2000.

[34] Robin Burke. Hybrid recommender systems: Survey and experiments. *UMUAI*, 12(4):331–370, 2002.

[35] Rocío Cañamares, Marcos Redondo, and Pablo Castells. Multi-armed recommender system bandit ensembles. In *RecSys*, 2019.

[36] Rich Caruana. Multitask learning. *MLJ*, 28(1):41–75, 1997.

[37] Rose Catherine and William Cohen. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *RecSys*, 2016.

[38] Rose Catherine, Kathryn Mazaitis, Maxine Eskenazi, and William Cohen. Explainable entity-based recommendations with knowledge graphs. *RecSys*, 2017.

[39] Antoine Channarond, Jean-Jacques Daudin, Stéphane Robin, et al. Classification and estimation in the stochastic blockmodel based on the empirical degrees. *EJS*, 6:2574–2601, 2012.

[40] Hanxiong Chen, Xu Chen, Shaoyun Shi, and Yongfeng Zhang. Generate natural language explanations for recommendation. 2019.

[41] Alex Chin, Yatong Chen, Kristen M. Altenburger, and Johan Ugander. Decoupled smoothing on graphs. In *WWW*, 2019.

[42] Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan. Domain adaptation of rule-based annotators for named-entity recognition tasks. In *EMNLP*, 2010.

[43] Peter Christen. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection.* Springer: Data-centric systems and applications, 2012.

[44] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. Computational fact checking from knowledge networks. *PloS one*, 10(6), 2015.

[45] Diane J Cook and Lawrence B Holder. *Mining graph data.* John Wiley & Sons Inc., 2006.

[46] Pranav Dandekar, Ashish Goel, and David Lee. Biased assimilation, homophily, and the dynamics of polarization. In *WINE*, 2012.

[47] Luc De Raedt and Kristian Kersting. Probabilistic inductive logic programming. In *PILP*. 2008.

[48] Luc De Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100:5–47, 2015.

[49] Luc De Raedt, Anton Dries, Ingo Thon, Guy Van den Broeck, and Mathias Verbeke. Inducing probabilistic relational rules from probabilistic examples. In *IJCAI*, 2015.

[50] Luc De Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neuro-symbolic artificial intelligence. *IJCAI*, 2020.

[51] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[52] AnHai Doan and Alon Y Halevy. Semantic integration research in the database community: A brief survey. *AI magazine*, 2005.

[53] Pedro Domingos. Multi-relational record linkage. In *KDD Workshop on MRMD*, 2004.

[54] Xin Dong, Alon Halevy, and Jayant Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.

[55] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.

[56] Xin Luna Dong. Challenges and innovations in building a product knowledge graph. In *KDD*, pages 2869–2869. ACM, 2018.

[57] Xin Luna Dong, Alon Halevy, and Cong Yu. Data integration with uncertainty. In *VLDB*, 2009.

[58] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. Knowledge-based trust: Estimating the trustworthiness of web sources. *VLDB*, 2015.

[59] Simon Dooms. Dynamic generation of personalized hybrid recommender systems. In *RecSys*, 2013.

[60] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *Springer Challenges in ML*, 2017.

[61] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. Entity disambiguation for knowledge base population. In *International Conference on Computational Linguistics (ICCL)*, pages 277–285. Association for Computational Linguistics, 2010.

[62] Cody Dunne and Ben Shneiderman. Motif simplification: improving network visualization readability with fan, connector, and clique glyphs. In *CHI*, 2013.

[63] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. Deeper–deep entity resolution. In *VLDB*, 2017.

[64] Javid Ebrahimi, Dejing Dou, and Daniel Lowd. Weakly supervised tweet stance classification by relational bootstrapping. In *EMNLP*, 2016.

[65] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. Collaborative filtering recommender systems. *FTHCI*, 4(2):81–173, 2011.

[66] Ahmed El-Kishky, Yanglei Song, Chi Wang, Clare R Voss, and Jiawei Han. Scalable topical phrase mining from text corpora. In *VLDB*, 2014.

[67] Mehdi Elahi, Francesco Ricci, and Neil Rubens. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*, 20:29–50, 2016.

[68] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. Duplicate record detection: A survey. *TKDE*, 2007.

[69] Varun Embar, Dhanya Sridhar, Golnoosh Farnadi, and Lise Getoor. Scalable structure learning for probabilistic soft logic. In *StarAI*, 2018.

[70] Varun Embar, Sriram Srinivasan, and Lise Getoor. Tractable marginal inference for hinge-loss markov random fields. In *ICML Workshop on TPM*, 2019.

[71] Varun Embar, Bunyamin Sisman, Hao Wei, Xin Luna Dong, Christos Faloutsos, and Lise Getoor. Contrastive entity linkage: Mining variational attributes from large catalogs for entity linkage. In *AKBC*, 2020.

[72] Varun Embar, Andrey Kan, Bunyamin Sisman, Christos Faloutsos, and Lise Getoor. Diffxtract: Joint discriminative product attribute-value extraction. 2021.

[73] Varun Embar, Sriram Srinivasan, and Lise Getoor. A comparison of statistical relational learning and graph neural networks for aggregate graph queries. *Machine Learning*, 110:1847–1866, 07 2021.

[74] Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. Reasoning about record matching rules. In *VLDB*, 2009.

[75] Ivan P Fellegi and Alan B Sunter. A theory for record linkage. *JASA*, 64: 1183–1210, 1969.

[76] Peter Forbes and Mu Zhu. Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation. In *RecSys*, 2011.

[77] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5): 75–174, 2010.

[78] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, 1999.

[79] Mohamed H Gad-Elrab, Daria Stepanova, Jacopo Urbani, and Gerhard Weikum. Exception-enriched rule learning from knowledge graphs. In *ISWC*, 2016.

[80] Mohamed H Gad-Elrab, Daria Stepanova, Trung-Kien Tran, Heike Adel, and Gerhard Weikum. Excut: explainable embedding-based clustering over knowledge graphs. In *International Semantic Web Conference*, pages 218–237. Springer, 2020.

[81] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. Fast rule mining in ontological knowledge bases with amie++. *VLDB*, 24 (6):707–730, 2015.

[82] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, 2013.

[83] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*, 2010.

[84] Yuqing Gao, Jisheng Liang, Benjamin Han, Mohamed Yakout, and Ahmed Mohamed. Building a large-scale, accurate and fresh knowledge graph. *KDD Tutorial*, 2018.

[85] Matt Gardner and Tom Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *EMNLP*, 2015.

[86] Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. Improving learning and inference in a large knowledge-base using latent syntactic cues. 2013.

[87] Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. In *EMNLP*, 2014.

[88] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.

[89] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning.* The MIT Press, 2007.

[90] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. Text mining for product attribute extraction. *KDD Explorations Newsletter*, 2006.

[91] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice.* Chapman and Hall/CRC, 1995.

[92] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

[93] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F Naughton, Narasimhan Rampalli, Jude Shavlik, and Xiaojin Zhu. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*, 2014.

[94] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[95] Asela Gunawardana and Christopher Meek. A unified approach to building hybrid recommender systems. In *RecSys*, 2009.

[96] Rishi Gupta, Tim Roughgarden, and Comandur Seshadhri. Decompositions of triangle-dense graphs. In *ITCS*, 2014.

[97] Ido Guy, Naama Zwerdling, Inbal Ronen, David Carmel, and Erel Uziel. Social media recommendation based on people and tags. In *SIGIR*, 2010.

[98] Alon Y Halevy, Oren Etzioni, AnHai Doan, Zachary G Ives, Jayant Madhavan, Luke K McDowell, and Igor Tatarinov. Crossing the structure chasm. In *CIDR*, 2003.

[99] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeuRIPS*, 2017.

[100] L Vivek Harsha Vardhan, Guo Jia, and Stanley Kok. Probabilistic logic graph attention networks for reasoning. In *WWW Companion*, 2020.

[101] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*, 2015.

[102] Reinhard Heckel, Michail Vlachos, Thomas Parnell, and Celestine Dünner. Scalable and interpretable product recommendations via overlapping co-clustering. In *ICDE*, 2017.

[103] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *CSCW*, 2000.

[104] Rob High. The era of cognitive systems: An inside look at ibm watson and how it works. *IBM Corporation Redbook*, 1:16, 2012.

[105] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[106] Pascal Hitzler, Markus Krotzsch, and Sebastian Rudolph. *Foundations of semantic web technologies*. Chapman and Hall/CRC, 2009.

[107] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Soc. Netw.*, 5:109 – 137, 1983.

[108] Andrea Horch, Holger Kett, and Anette Weisbecker. Matching product offers of e-shops. In *PAKKD*, 2016.

[109] Yunfeng Hou, Ning Yang, Yi Wu, and S Yu Philip. Explainable recommendation with fusion of aspect information. *WWW*, 22(1):221–240, 2019.

[110] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *WSDM*, 2019.

[111] Tuyen N. Huynh and Raymond J. Mooney. Discriminative structure and parameter learning for Markov logic networks. In *ICML*, 2008.

[112] Tuyen N Huynh and Raymond J Mooney. Online structure learning for markov logic networks. In *ECML*, 2011.

[113] Robert Isele, Anja Jentzsch, and Christian Bizer. Silk server-adding missing links while consuming linked data. In *COLD*, 2010.

[114] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[115] Kristen Johnson and Dan Goldwasser. "All I know about politics is what I read in Twitter": Weakly supervised models for extracting politicians' stances from Twitter. In *COLING*, 2016.

[116] Anitha Kannan, Inmar E Givoni, Rakesh Agrawal, and Ariel Fuxman. Matching unstructured product offers to structured product specifications. In *KDD*, 2011.

[117] Giannis Karamanolakis, Jun Ma, and Xin Luna Dong. Txtract: Taxonomy-aware knowledge extraction for thousands of product categories. *KDD*, 2020.

[118] Kristian Kersting and Luc De Raedt. Bayesian logic programming: Theory and tool. *, An introduction to Statistical Relational Learning*, 2007.

[119] Hassan Khosravi, Oliver Schulte, Tong Man, Xiaoyuan Xu, and Bahareh Bina. Structure learning for Markov logic networks with many descriptive attributes. In *AAAI*, 2010.

[120] Tushar Khot, Sriraam Natarajan, Kristian Kersting, and Jude Shavlik. Learning markov logic networks via functional gradient boosting. In *ICDM*, 2011.

[121] Sung Guen Kim and Juyoung Kang. Analyzing the discriminative attributes of products using text mining focused on cosmetic reviews. *Information Processing & Management*, 54(6):938–957, 2018.

[122] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[123] Stanley Kok and Pedro Domingos. Learning Markov logic network structure via hypergraph lifting. In *ICML*, 2009.

[124] Stanley Kok and Pedro Domingos. Learning Markov logic networks using structural motifs. In *ICML*, 2010.

[125] Stanley Kok, Marc Sumner, Matthew Richardson, Parag Singla, Hoifung Poon, Daniel Lowd, Jue Wang, and Pedro Domingos. The alchemy system for statistical relational ai. Technical report, UW, Seattle, 2005.

[126] Daphne Koller. Probabilistic relational models. In *ILP*, 1999.

[127] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[128] Pradap Konda, Sanjib Das, Paul Suganthan GC, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, et al. Magellan: Toward building entity matching management systems. In *VLDB*, 2016.

[129] Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 2010.

[130] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of entity resolution approaches on real-world match problems. In *VLDB*, 2010.

[131] Hanna Köpcke, Andreas Thor, Stefan Thomas, and Erhard Rahm. Tailoring entity resolution for matching product offers. In *EDBT*, 2012.

[132] Nick Koudas, Sunita Sarawagi, and Divesh Srivastava. Record linkage: similarity measures and algorithms. In *SIGMOD*, 2006.

[133] Pigi Kouki, Shobeir Fakhraei, James Foulds, Magdalini Eirinaki, and Lise Getoor. Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In *RecSys*, 2015.

[134] Pigi Kouki, James Schaffer, Jay Pujara, John O'Donovan, and Lise Getoor. Personalized explanations for hybrid recommender systems. In *IUI*, 2019.

[135] Zornitsa Kozareva, Qi Li, Ke Zhai, and Weiwei Guo. Recognizing salient entities in shopping queries. In *ACL*, 2016.

[136] Alicia Krebs, Alessandro Lenci, and Denis Paperno. Semeval-2018 task 10: Capturing discriminative attributes. In *SEMEVAL*, 2018.

[137] Veton Këpuska and Gamal Bohouta. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *CCWC*, 2018.

[138] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*, 2001.

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

[140] Ni Lao, Tom Mitchell, and William W Cohen. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, 2011.

[141] Kristen LeFevre and Evimaria Terzi. Grass: Graph structure summarization. In *ICDM*, 2010.

[142] Douglas B Lenat, Ramanathan V. Guha, Karen Pittman, Dexter Pratt, and Mary Shepherd. Cyc: toward programs with common sense. *Communications of the ACM*, 33(8):30–49, 1990.

[143] Cheng-Te Li and Shou-De Lin. Egocentric information abstraction for heterogeneous social networks. In *ASONAM*, 2009.

[144] Manling Li, Qi Zeng, Ying Lin, Kyunghyun Cho, Heng Ji, Jonathan May, Nathanael Chambers, and Clare Voss. Connecting the dots: Event graph schema induction with path language modeling. In *EMNLP*, pages 684–695, 2020.

[145] Pei Li, Xin Luna Dong, Songtao Guo, Andrea Maurino, and Divesh Srivastava. Robust group linkage. In *WWW*, 2015.

[146] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. Neural rating regression with abstractive tips generation for recommendation. In *SIGIR*, 2017.

[147] Guang Ling, Michael R Lyu, and Irwin King. Ratings meet reviews, a combined approach to recommend. In *RecSys*, 2014.

[148] Xiao Ling and Daniel S Weld. Fine-grained entity recognition. In *AAAI*, 2012.

[149] Hugo Liu and Push Singh. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.

[150] Juntao Liu, Caihua Wu, and Wenyu Liu. Bayesian probabilistic matrix factorization with social relations and item contents for recommendation. *DSS*, 55(3):838–850, 2013.

[151] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. Graph summarization methods and applications: A survey. *CSUR*, 51:62–96, 2018.

[152] Robert L Logan IV, Samuel Humeau, and Sameer Singh. Multimodal attribute extraction. *AKBC*, 2017.

[153] Nikhil Londhe, Vishrawas Gopalakrishnan, Aidong Zhang, Hung Q Ngo, and Rohini Srihari. Matching titles with cross title web-search enrichment and community detection. In *VLDB*, 2014.

[154] Ben London, Bert Huang, and Lise Getoor. Stability and generalization in structured prediction. *Journal of Machine Learning Research*, 17, 2016. to appear.

[155] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[156] Mi Luo, Fei Chen, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Jiashi Feng, and Zhenguo Li. Metaselector: Meta-learning for recommendation with user-level adaptive model selection. In *WebConf*, 2020.

[157] Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.

[158] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *WSDM*, 2011.

[159] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, 2013.

[160] Brian McFee, Thierry Bertin-Mahieux, Daniel PW Ellis, and Gert RG Lanckriet. The million song dataset challenge. In *WWW*, 2012.

[161] Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. Semantic feature production norms for a large set of living and nonliving things. *Behavior research methods*, 37(4):547–559, 2005.

[162] Lilyana Mihalkova and Raymond J Mooney. Bottom-up learning of Markov logic network structure. In *ICML*, 2007.

[163] Andrei Mikheev, Marc Moens, and Claire Grover. Named entity recognition without gazetteers. In *EACL*, 1999.

[164] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[165] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[166] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, B Yang, J Betteridge, A Carlson, B Dalvi, M Gardner, B Kisiel, et al. Never-ending learning. *Communications of the ACM*, 61(5):103–115, 2018.

[167] Vassil Momtchev, Deyan Peychev, Todor Primov, and Georgi Georgiev. Expanding the pathway and interaction knowledge in linked life data. *ISWC*, 2009.

[168] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In *SIGMOD*, 2018.

[169] Stephen Muggleton. Inductive logic programming. *New generation computing*, 8(4):295–318, 1991.

[170] Stephen Muggleton et al. Stochastic logic programs. *Advances in inductive logic programming*, 32:254–264, 1996.

[171] Katarzyna Musiał and Krzysztof Juszczyszyn. Properties of bridge nodes in social networks. In *ICCCI*, 2009.

[172] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30:3–26, 2007.

[173] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. Graph summarization with bounded error. In *MOD*, 2008.

[174] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[175] Jennifer Neville and David Jensen. Iterative classification in relational data. In *AAAI Workshop on Learning Statistical Models from Relational Data*, 2002.

[176] Jennifer Neville and David Jensen. Dependency networks for relational data. In *ICDM*, 2004.

[177] Jennifer Neville and David Jensen. Relational dependency networks. *JMLR*, 8:653–692, 2007.

[178] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[179] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[180] Feng Niu, Christopher Ré, AnHai Doan, and Jude Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *VLDB*, 4: 373–384, 2011.

[181] Byung-Won On, Nick Koudas, Dongwon Lee, and Divesh Srivastava. Group linkage. In *ICDE*, 2007.

[182] Jeff Z Pan, Siyana Pavlova, Chenxi Li, Ningxi Li, Yangmei Li, and Jinshuo Liu. Content based fake news detection using knowledge graphs. In *ISWC*, 2018.

[183] Denis Parra, Peter Brusilovsky, and Christoph Trattner. See what you want to see: visual user-driven approach for hybrid recommendation. In *IUI*, 2014.

[184] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 325–341. Springer, 2007.

[185] Georgina Peake and Jun Wang. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *SIGKDD*, 2018.

[186] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[187] Petar Petrovski and Christian Bizer. Extracting attribute-value pairs from product specifications on the web. In *WI*, 2017.

[188] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Column networks for collective classification. In *AAAI*, 2017.

[189] Emmanouil Platanios, Hoifung Poon, Tom M Mitchell, and Eric J Horvitz. Estimating accuracy from unlabeled data: A probabilistic logic approach. In *NeuRIPS*, 2017.

[190] Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, 2006.

[191] Jay Pujara. *Probabilistic models for scalable knowledge graph construction*. PhD thesis, 2016.

[192] Jay Pujara and Lise Getoor. Generic statistical relational entity resolution in knowledge graphs. *StaRAI*, 2016.

[193] Jay Pujara, Kevin Murphy, Xin Luna Dong, and Curtis Janssen. Probabilistic models for collective entity resolution between knowledge graphs. In *Bay Area Machine Learning Symposium*, 2014.

[194] Duangmanee Pew Putthividhya and Junling Hu. Bootstrapped named entity recognition for product attribute extraction. In *EMNLP*, 2011.

[195] Lin Qiu, Sheng Gao, Wenlong Cheng, and Jun Guo. Aspect-based latent factor model by integrating ratings and reviews for recommender system. *Know.-Based Syst.*, 110(C):233–243, 2016.

[196] Meng Qu and Jian Tang. Probabilistic logic neural networks for reasoning. In *NeuRIPS*, 2019.

[197] Meng Qu, Yoshua Bengio, and Jian Tang. Gmnn: Graph markov neural networks. In *ICML*, 2019.

[198] Qiang Qu, Siyuan Liu, Christian S Jensen, Feida Zhu, and Christos Faloutsos. Interestingness-driven diffusion process summarization in dynamic networks. In *ECML*, 2014.

[199] Luc De Raedt and Kristian Kersting. Statistical relational learning. In *EML*. Springer, 2011.

[200] Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. *Statistical relational artificial intelligence: Logic, probability, and computation*. Morgan & Claypool Publishers, 2016.

[201] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[202] Vibhor Rastogi, Nilesh Dalvi, and Minos Garofalakis. Large-scale collective entity matching. *VLDB*, 4(4):208–218, 2011.

[203] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[204] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *SIGKDD*, 2016.

[205] Matthew Richardson and Pedro Domingos. Markov logic networks. *MLJ*, 62(1-2):107–136, 2006.

[206] Petar Ristoski, Petar Petrovski, Peter Mika, and Heiko Paulheim. A machine learning approach for product matching and categorization. *SWJ*, 9: 1–22, 2017.

[207] Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *NeuRIPS*. 2017.

[208] Rahmtin Rotabi, Krishna Kamath, Jon Kleinberg, and Aneesh Sharma. Detecting strong ties using network motifs. In *World Wide Web Companion*, pages 983–992, 2017.

[209] Alieh Saeedi, Eric Peukert, and Erhard Rahm. Using link features for entity clustering in knowledge graphs. In *European Semantic Web Conference*, pages 576–592. Springer, 2018.

[210] Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *AAAI*, 2018.

[211] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *KDD*, 2002.

[212] Ruhi Sarikaya, Geoffrey E Hinton, and Anoop Deoras. Application of deep belief networks for natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):778–784, 2014.

[213] Sandeepkumar Satpal, Sahely Bhadra, Sundararajan Sellamanickam, Rajeev Rastogi, and Prithviraj Sen. Web information extraction using markov logic networks. In *KDD*, 2011.

[214] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 291–324. Springer, 2007.

[215] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

[216] John Scott. Social network analysis. *Sociology*, 22:109–127, 1988.

[217] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29:93–93, 2008.

[218] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *RecSys*, 2017.

[219] Amit Sharma and Dan Cosley. Do social explanations work? studying and modeling the effects of social explanations in recommender systems. In *WWW*, 2013.

[220] Zeqian Shen, Kwan-Liu Ma, and Tina Eliassi-Rad. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *TVCG*, 12:1427–1439, 2006.

[221] Lei Shi, Hanghang Tong, Jie Tang, and Chuang Lin. Vegas: Visual influence graph summarization on citation networks. *TKDE*, 27:3417–3431, 2015.

[222] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *ICML*, 2017.

[223] Rohit Singh, Vamsi Meduri, Ahmed Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. Generating concise entity matching rules. In *ICDM*, 2017.

[224] Amit Singhal. Introducing the knowledge graph: things, not strings. *Official google blog*, 2012.

[225] Parag Singla and Pedro Domingos. Entity resolution with markov logic. In *ICDM*, pages 572–582. IEEE, 2006.

[226] Pia Sommerauer and Antske Fokkens. Firearms and tigers are dangerous, kitchen knives and zebras are not: Testing whether word embeddings can tell. In *EMNLP Workshop on BlackboxNLP*, 2018.

[227] Armins Stepanjans and André Freitas. Identifying and explaining discriminative attributes. In *EMNLP*, 2019.

[228] Michael Stonebraker, Daniel Bruckner, Ihab F Ilyas, George Beskales, Mitch Cherniack, Stanley B Zdonik, Alexander Pagan, and Shan Xu. Data curation at scale: The data tamer system. In *CIDR*, 2013.

[229] David G Stork, Richard O Duda, Peter E Hart, and D Stork. *Pattern classification.* Wiley-Interscience Publication, 2001.

[230] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *TheWebConf*, 2007.

[231] Charles Sutton and Andrew McCallum. Piecewise pseudolikelihood for efficient training of conditional random fields. In *ICML*, 2007.

[232] Aaron Swartz. Musicbrainz: A semantic web service. *IEEE Intelligent Systems*, 17:76–77, 2002.

[233] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearson Education, 2006.

[234] Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. 2002.

[235] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *ICDEW*, 2007.

[236] Nava Tintarev and Judith Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender systems handbook*, pages 479–510. Springer, 2011.

[237] Son D Tran and Larry S Davis. Event modeling and recognition using markov logic networks. In *ECCV*, 2008.

[238] Rakshit Trivedi, Bunyamin Sisman, Jun Ma, Christos Faloutsos, Hongyuan Zha, and Xin Luna Dong. Linknbed: Multi-graph representation learning with entity linkage. In *ACL*, 2018.

[239] Chun-Hua Tsai and Peter Brusilovsky. Explaining recommendations in an interactive hybrid social recommender. In *IUI*, 2019.

[240] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[241] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018.

[242] D Venugopal, S Sarkhel, and V Gogate. Magician: Scalable inference and learning in markov logic using approximate symmetries. Technical report, UofM, Memphis, 2016.

[243] Jesse Vig, Shilad Sen, and John Riedl. Tagsplanations: explaining recommendations using tags. In *IUI*, 2009.

[244] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR 2001*, 2001.

[245] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. In *VLDB*, 2012.

[246] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. Explainable recommendation via multi-task learning in opinionated text data. In *SIGIR*, 2018.

[247] Qifan Wang, Li Yang, Bhargav Kanagal, Sumit Sanghai, D Sivakumar, Bin Shu, Zac Yu, and Jon Elsas. Learning to extract attribute value from product via question answering: A multi-task approach. In *KDD*, 2020.

[248] Quan Wang, Jing Liu, Yuanfei Luo, Bin Wang, and Chin-Yew Lin. Knowledge base completion via coupled path ranking. In *ACL*, volume 1, pages 1308–1318, 2016.

[249] William Yang Wang and William Cohen. Joint information extraction and reasoning: A scalable statistical relational learning approach. In *IJCNLP*, 2015.

[250] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[251] Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. Nlprolog: Reasoning with weak unification for question answering in natural language. *ACL*, 2019.

[252] Lior Wolf, Tomer Galanti, and Tamir Hazan. A formal approach to explainability. In *AAAI/ACM Conference on AI, Ethics, and Society*, pages 255–261, 2019.

[253] Yao Wu and Martin Ester. Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *WSDM*, 2015.

[254] Ye Wu, Zhinong Zhong, Wei Xiong, and Ning Jing. Graph summarization for attributed graphs. In *ISEEE*, 2014.

[255] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[256] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Yu Philip S. A comprehensive survey on graph neural networks. volume 32, pages 4–24, 2021.

[257] Huimin Xu, Wenting Wang, Xinnian Mao, Xinyu Jiang, and Man Lan. Scaling up open tagging from tens to thousands: Comprehension empowered attribute value extraction from product title. In *ACL*, 2019.

[258] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *ICLR*, 2015.

[259] Fan Yang, Ninghao Liu, Suhang Wang, and Xia Hu. Towards interpretation of recommender systems with sorted explanation paths. In *ICDM*, 2018.

[260] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

[261] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*, 2014.

[262] Yanlei Yu, Zhiwu Lu, Jiajun Liu, Guoping Zhao, and Ji-rong Wen. Rum: Network representation learning using motifs. 2019.

[263] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *IN ECCV*, 2014.

[264] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *Found Trends Inf. Ret.*, 14(1):1–101, 2018.

[265] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*, 2014.

[266] Yue Zhang and Arti Ramesh. Learning interpretable relational structures of hinge-loss markov random fields. In *IJCAI*, 2019.

[267] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. Efficient probabilistic logic reasoning with graph neural networks. 2020.

[268] Kaiqi Zhao, Gao Cong, Quan Yuan, and Kenny Q Zhu. Sar: A sentiment-aspect-region model for user preference analysis in geo-tagged reviews. In *ICDE*, 2015.

[269] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. Opentag: Open attribute value extraction from product profiles. In *KDD*, 2018.