Learning Logical Rules from Knowledge Graphs

Yulong Gu

School of Computing
Newcastle University

In Partial Fulfilment of the Requirements for the Degree of
Doctor of Philosophy (Integrated)

November 2021

Acknowledgements

The journey to complete the works in this thesis has been full of obstacles and enjoyments. Without the support and love from those who accompanied me throughout this course, it is certainly impossible for me to finish this thesis. I would like to express my deepest gratitude to my supervisor Paolo Missier, for his wise guidance; for enduring many of my naive ideas; for patiently reviewing my not-so-ready drafts, and for always being positive and supportive, which is truly precious to me. Thank you Paolo. I also want to thank my co-supervisor Yu Guan, with whom I shared many valuable and insightful conversations. I would like to thank my dear friends, Bowen Li, Jiajie Zhang, Yinhao Li and Zongying Yang, for all the merry moments that fueled me up to keep going and for all the constructive discussions about my work. Lastly, I would like to thank my girlfriend Huan Wu, who has cruised the entirety of this journey with me, for being the light even in the darkest night. To my parents, Jian Gu and Chunmei Zhang, thanks for being my motivation and my refuge.

Abstract

Expressing and extracting regularities in multi-relational data, where data points are interrelated and heterogeneous, requires well-designed knowledge representation. Knowledge Graphs (KGs), as a graph-based representation of multi-relational data, have seen a rapidly growing presence in industry and academia, where many real-world applications and academic research are either enabled or augmented through the incorporation of KGs. However, due to the way KGs are constructed, they are inherently noisy and incomplete. In this thesis, we focus on developing logic-based graph reasoning systems that utilize logical rules to infer missing facts for the completion of KGs. Unlike most rule learners that primarily mine abstract rules that contain no constants, we are particularly interested in learning instantiated rules that contain constants due to their ability to represent meaningful patterns and correlations that can not be expressed by abstract rules. The inclusion of instantiated rules often leads to exponential growth in the search space. Therefore, it is necessary to develop optimization strategies to balance between scalability and expressivity. To such an end, we propose GPFL, a probabilistic rule learning system optimized to mine instantiated rules through the implementation of a novel two-stage rule generation mechanism. Through experiments, we demonstrate that GPFL not only performs competitively on knowledge graph completion but is also much more efficient then existing methods at mining instantiated rules. With GPFL, we also reveal overfitting instantiated rules and provide detailed analyses about their impact on system performance. Then, we propose RHF, a generic framework for constructing rule hierarchies from a given set of rules. We demonstrate through experiments that with RHF and the hierarchical pruning techniques enabled by it, significant reductions in runtime and rule size are observed due to the pruning of unpromising rules. Eventually, to test the practicability of rule learning systems, we develop Ranta, a novel drug repurposing system that relies on logical rules as features to make interpretable inferences. Ranta outperforms existing methods by a large margin in predictive performance and can make reasonable repurposing suggestions with interpretable evidence.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

With the recent explosion of data and the availability of more powerful ubiquitous computing hardware, intelligent systems that aid humans' daily lives from many perspectives are enabled by Machine Learning approaches that make accurate predictions through the learning of the data prepared in various forms. To address the pressing problem of information fusion [86] which is to integrate data of different types from multiple sources for intelligent applications, Knowledge Graphs (KGs) [33], as a graph-structured representation of knowledge, have gained massive attention in that interrelated heterogeneous data points can be intuitively represented and efficiently queried. In recent years, many public and private KGs have been created to support downstream tasks, including information retrieval [112], question answering [28, 12] and recommender systems [143, 131]. Open-source KG curation projects, such as YAGO [56], DBpedia [4] and Wikidata [126], that aim to create collections of universal knowledge to enable a wide range of applications, are welcomed by both industry and academia.

However, since the construction of large-scale KGs often involves some degree of automation, the resulting KGs are inherently incomplete and noisy. Besides, the intentional definitions of components in KGs that represent general knowledge about the domain of discourse are often not provided. To address both of these issues, knowledge graph reasoning systems, that formulate the identification of errors and inference of new facts as the link prediction problem where plausibility scores are estimated for the links in KGs, have seen rapid growth in recent years [129]. There are three classes of knowledge graph reasoning system from the perspective of knowledge representation and learning mechanism. Distributed Representation (DR) methods utilize geometric constraints to project the components of KGs onto a low-dimensional embedding space. The generated embeddings can then be used for graph analytics, clustering and classification. Graph Neural Networks (GNNs) [146] are designed to provide end-to-end solutions to specific tasks. The specialized architectures of GNN variants usually utilize message passing and neighbor aggregation to handle the irregularity and the lack of topology issues appearing in graph-structured data models. Though both DR and GNN models demonstrate competitive predictive power on link prediction tasks, their inability to provide interpretable insights about prediction results prevents their application to many risk-sensitive real-world scenarios, such as healthcare insurance and biomedical research. The logic-based methods [39, 91, 75] that induce high-level regularities in the form of first-order logic rules from KGs are considered one of the most promising solutions to the lack of interpretability issue. Logical rule learning systems produce the high-quality rules that can be used to make inductive inference for filling in missing links or provide interpretable insights about the underlying patterns in KGs.

In addition to the choice of implementation strategies, rule learning systems require three mechanically critical components to function. First, language bias dictates what types of rules to learn. In particular, given a language bias, a well-formed rule must conform to the syntactic and semantic constraints specified to balance between scalability and expressivity because complex rules, e.g., rules containing constants, are more expressive and the allowance of their creation often exponentially enlarges the search space, thus the system becomes less scalable. Second, problem definition decides the success conditions required to find a solution rule set. For instance, classic Inductive Logic Programming (ILP) works [82] employ a strict rule discovery problem definition where the rules in the solution set must cover all of the positive and none of the negative examples. In contrast, most recent works take an approximate strategy that allows the coverage of negative examples in response to the inevitable noises in KGs. Third, the learning paradigm details the specific procedure with which the rules are generated. For a rule learning system, both the problem definition and learning paradigm must be designed to efficiently fulfil the language bias to make the system practical in real-world applications. As demonstrated in recent works [76, 75], utilizing a language bias that allows instantiated logical rules containing constants is significantly beneficial to systems' predictive performance and interpretability. However, the development of systems optimizing the mining of instantiated rules and the studies of such systems' characteristics and applications are not well-researched in the community. This is because the benefit of instantiated rules has only been discovered recently, and the rule learning community is relatively small.

This thesis aims to develop efficient strategies to enable the optimized learning of instantiated logical rules and explore the learned rules' characteristics and possible applications. To such an end, we first propose a novel rule learning system that utilizes a two-stage learning paradigm to mine instantiated rules efficiently. Through experiments, we observed that instantiated rules are more prone to overfitting due to being more specific than the abstract rules that contain no constants. Therefore, we conducted a detailed analysis of the overfitting issue with the proposed rule learning system. Then, we develop a generic optimization strategy that uses the inherent subsumption relationships between rules to identify and prune irrelevant and redundant rules in order to achieve better performance. Eventually, we explore the application of rule learning systems to a crucial real-world problem, the drug repurposing problem, to test the systems' practicability. We design a feature engineering based system that selects a minimal set of instantiated rules and treats the learned rules as features to generate a feature matrix for the training of a drug-efficacy model.

1.1 Contribution

This thesis is composed of the following three original works:

Graph Path Feature Learning In this work, we present GPFL, a probabilistic rule learning system optimized to mine instantiated first-order logic rules from KGs. Instantiated rules contain constants extracted from KGs. Compared to abstract rules that contain no constants, instantiated rules are capable of explaining and expressing concepts in more detail. GPFL utilizes a novel two-stage rule generation mechanism that first generalizes extracted paths into templates that are acyclic abstract rules until a certain degree of template saturation is achieved, then specializes the generated templates into instantiated rules. Unlike existing works that ground every mined instantiated rule for evaluation, GPFL shares groundings between structurally similar rules for collective evaluation. Moreover, we reveal the presence of overfitting rules, their impact on predictive performance, and the effectiveness of a simple validation method filtering out overfitting rules. Through extensive experiments on public benchmark datasets, we show that GPFL 1.) significantly reduces the runtime on evaluating instantiated rules; 2.) discovers many more quality instantiated rules than existing works; 3.) improves the predictive performance of learned rules by removing overfitting rules via validation; 4.) is competitive on the knowledge graph completion task compared to state-of-the-art baselines.

Rule Hierarchy Framework In this work, we aim to address the scalability issue introduced by the generation and evaluation of unpromising rules with bottom-up rule learning systems. Most of the existing bottom-up systems use Flat Pruning Methods (FPMs) that filter out low-quality rules by checking against pre-defined quality thresholds. Since the rule hierarchies that contain subsumption relationships between rules are often not readily available in bottom-up systems, the more effective Hierarchical Pruning Methods (HPMs) employed by top-down systems can not be applied directly to benefit bottom-up systems. We introduce a generic Rule Hierarchy Framework (RHF) that leverages a collection of novel subsumption frameworks to build proper rule hierarchies from the rules produced by bottom-up learners. Then, the rule hierarchies can enable the application of HPMs to bottom-up systems for the better pruning of unpromising rules. As a case study, we adapt RHF and two HPMs to GPFL and conduct extensive experiments on four public benchmark datasets. We show that the application of HPMs effectively removes unpromising rules, leading to significant reductions in the runtime and the number of learned rules, without compromising predictive performance.

Learning Logical Rules for Drug Repurposing In this work, we aim to explore the applicability of rule learning systems on the drug repurposing task. We propose Ranta, a novel drug repurposing system that takes a logic-based feature engineering strategy. The drug repurposing task is formulated as a probabilistic binary classification problem where a conventional machine learning model learns to predict with confidence whether a drug has indication for a disease. We take four steps to achieve this: 1.) a base rule learner is employed to extract both abstract and instantiated logical rules from biomedical KGs; 2.) as the volume of instantiated rules is too large, we utilize a rule model tuner to select a minimal subset of instantiated rules as a rule model that best represents the learning target; 3.) a feature matrix is constructed where the columns are rules, rows are examples and cell values are relevance scores, and 4.) the feature matrix is passed to train a logistic regression model for making drug repurposing inferences. Through extensive experiments on two large-scale biomedical KGs, we show that Ranta significantly outperforms existing works in predictive performance. Due to the approximation strategies employed to compute cell values, Ranta also runs at most 10 times faster than a state-of-the-art drug repurposing system. With a case study where we repurpose drugs for Pulmonary Arterial Hypertension, we show that Ranta can make reasonable suggestions and provide interpretable evidence in the form of logical rules.

The following publications and manuscripts have resulted from doing the research introduced in this thesis:

- **Gu, Y.**, and Missier, P. (2017). Adaptive incremental learning for statistical relational models using gradient-based boosting. In 27th International Conference on Inductive Logic Programming.
- **Gu, Y.**, Guan, Y., and Missier, P. (2020). Building Rule Hierarchies for Efficient Logical Rule Learning from Knowledge Graphs. arXiv preprint arXiv:2006.16171.
- **Gu, Y.**, Guan, Y., and Missier, P. (2020). Towards Learning Instantiated Logical Rules from Knowledge Graphs. arXiv preprint arXiv:2003.06071.
- **Gu, Y.**, Skelton, J., Missier, P., & Wipat, A. (2021). Mining Logical Rules for Computational Drug Repurposing. *(to be submitted)*

1.2 Thesis Structure

This thesis is organized as follows: in Chapter 2, we introduce the fundamentals about KGs where we provide background information on why KGs are inherently incomplete and noisy and how to mitigate these issues. Then, by breaking up rule learning systems into three components, we provide an analysis on the characteristics of various existing rule learning systems; in Chapter 3, we introduce the design idea about the GPFL system and provide a study on the effects of overfitting rules on system performance; in Chapter 4, we introduce RHF in detail; in Chapter 5, we first provide an introduction about the drug repurposing problem and then focus on detailing the Ranta system and its experiments; in Chapter 6, we provide a conclusion about the work in this thesis and a discussion about future works.

Chapter 2

Background

This chapter provides the background information necessary to understand the methodologies proposed in the rest of this thesis. We start by giving a review on the fundamentals about Knowledge Graphs (KGs), where we discuss why KGs are inherently incomplete and noisy through the lens of KG construction and how these issues can be addressed by the graph reasoning algorithms that fill in missing links and discard contradictory facts. Then, by breaking up rule learning systems into three mechanically important parts, namely language bias, problem definition and learning paradigm, we provide an analysis of the characteristics of various existing systems.

2.1 Knowledge Graphs

In 2012, Google launched the Google Knowledge Graph, an intelligent data model that understands real-world entities and their relationships to one another [112]. In comparison to last generation search engines where search operations are mainly about matching users' queries to keywords, Google proposed to model the keywords that appeared in users' queries as entities and return not only the properties of the entities as search results but also information about other relevant entities. Along with Natural Language Processing (NLP) engines, this enrichment in information allows Google's augmented search engine to answer complex questions with much higher accuracy than before [94]. Since then, the idea of exploiting the structure of graphs to intuitively represent the domains involving interactions between entities has been attracting increasing attention from both industry and academia [33, 97, 93]. A Knowledge Graph (KG) is defined as a graph-based abstraction of knowledge where nodes represent entities and edges represent relations between these entities. For instance in Figure 2.1, the fact "Bill Gates speaks English" is represented by a triple $Speak(Bill\ Gates, English)$ where $Speak$ is the predicate, $Bill\ Gates$ the subject and $English$ the object. Over the last decade, many large-scale open-source KGs that contain common knowledge and real-world entities have been created, including NELL [78], Freebase [10], DBpedia [3], Wikidata [126] and YAGO [56], to support intelligent applications.

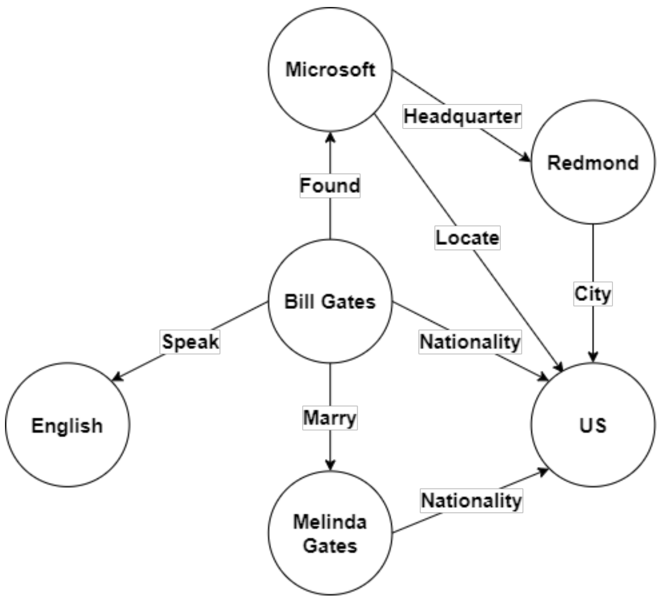


Fig. 2.1 A small knowledge graph where an example social network of the entity "Bill Gates" is presented.

A substantial number of works have been focusing on applying KGs to practical systems and specific domains in recent years [154]. For question answering [28], Bordes et al. [12] propose to learn low-dimensional embeddings for questions and Freebase's entities and predicates, and then compute a similarity score between the question and potential answers using the learned embeddings to suggest top answers. For recommender systems [143], Wang et al. [131] use a recurrent neural network leveraging sequential dependencies between entities and predicates within paths in a KG to model the underlying rationale of user-item interactions. In domain-specific applications, Qi et al. [98] built a cybersecurity KG which contains relationships between attacks, events and alarms and is used to predict cyberattacks through an association analysis algorithm. Liu et al. [70] constructed an enterprise KG by extracting company entities and their business relationships from news, where the items in the KG are later translated into embeddings and passed to a neural network for the prediction of the stock price. In particular, KGs have been applied extensively in biomedical fields. Ernst et al. [32] automatically integrate data from different public sources into a large-scale biomedical KG using advanced information extraction techniques to support various downstream tasks. Himmelstein et al. [52] employ similar method to build a KG that contains over 2 million drug and disease related facts, and use a feature engineering strategy to extract path features from the KG to repurpose drugs for diseases. Treating KGs as external knowledge to improve system performance is also a popular line of research. Zhang et al. [141] noticed that the semantic associations between scenes help humans to categorize images. Therefore, they build an image KG that encodes both real-world semantic and scene associations to aid convolutional neural networks for image classification. Ma et al. [71] propose to augment the LSTM network with a hierarchical attention mechanism that incorporates the commonsense knowledge represented in a KG to improve the system's performance on sentiment analysis tasks.

Among many aspects considered while evaluating the quality of KGs [33], completeness and accuracy determine the usefulness of a KG [23]. Completeness refers to the degree to which all the information of the target domain is encoded in the KG. To measure the completeness of a specific KG, an ideal KG is required encompassing all the elements that should appear in the KG in question. An ideal KG is often impossible to acquire due to the scale and evolution of the target domain. In practice, by comparing with gold standard samples and measuring the recall based on the facts inferred via meta-knowledge, a completeness score can be estimated. Accuracy measures the extent to which the information represented in the KG correctly reflects the corresponding concepts and facts in reality. Similar to the measurement of completeness, accuracy can be estimated through the comparison with labelled facts or the evaluation of extracted models. To further improve the performance of systems operating with KGs, it is important to understand why most large-scale KGs are inherently incomplete and noisy and how to mitigate the issues using reasoning techniques to complete and correct KGs.

2.1.1 Knowledge Graph Construction

Depending on the degree of human intervention and the types of source data, KG construction strategies can be categorized into four groups [87]. First, the KGs built with a manual curation strategy, such as Cyc [67], WordNet [77] and UMLS [9], contain triples that are produced by a closed group of experts. Second, the collaboratively curated KGs, such as Wikidata [126] and Freebase [11], are populated with triples contributed by volunteers through open knowledge curation projects. For instance, to allow contributors to edit facts in the Freebase KG in a collaborative manner, its developer Metaweb implemented a suite of tools, including a large data object store, a query language, a Web user interface and a lightweight typing system [33]. Besides the workloads required to build a collaborative platform, the dependence on human involvement makes the construction of KGs hard to scale. Studies [119] also suggest that the collaborative strategy is not sustainable due to increasing coordination and overhead costs. Though the accuracy of the hand-crafted triples is often very high, their inability to provide better coverage drives the community to develop automated strategies. The third group of methods automatically extract entities and relationships from semi-structured data sources. In particular, YAGO [56] and DBpedia [4] extract graph-structured data from Wikipedia where the formatted pages are processed into properties and entities. Finally, to incorporate the unstructured data on the Web, KGs, such as NELL [78], PROSPERA [83] and PATTY [84], are constructed using NLP methods to mine Web texts. Regardless of the choice of construction strategies, building a KG from scratch is non-trivial. It is hard to balance accuracy and completeness at the moment of construction in that the large-scale information extraction powered by advanced NLP algorithms is prone to generate erroneous facts given the dynamics of Web data and the process of fact validation often requires expensive, laborious work. Therefore, it is common to refine a KG after its construction by training a predictive model that encodes the underlying patterns of the KG [93].

2.1.2 Knowledge Graph Reasoning

Knowledge graph reasoning aims to identify errors and infer new conclusions from existing data [20]. Specifically, the task of identifying and removing existing false facts is known as Knowledge Graph Correction and the task to infer facts that are considered correct but not given by the KG is referred to as Knowledge Graph Completion. Algorithms for correction [108, 110, 114] normally assign plausibility scores to existing facts and filter out the facts with unsatisfactory score values. Similarly, given a set of potentially correct facts, completion algorithms [69, 109, 122] predict the probability of correctness for every candidate fact and only keep top-ranked ones. The shared problem formulation pattern between correction and completion systems, that is to estimate a plausibility score to support the removal or addition decision, coincides with the link prediction problem originally proposed in the area of Statistical Relational Learning [60]. Link prediction aims to predict whether an entity shares a specific relationship with another given entity. For instance in Figure 2.1, given a link prediction query $Speak(?, English)$ which is to predict who speaks English, a correction system will score the known facts $Speak(Bill\ Gates, English)$ for fact validation, whereas a completion system may try to predict the existence of a new fact $Speak(Melinda\ Gates, English)$ given that $Nationality(Melinda\ Gates, US)$. Therefore, knowledge graph reasoning can be reduced to the link prediction problem.

Link prediction systems based on Distributed Representation (DR), Graph Neural Networks (GNNs) and logical rules are among the most popular in the community. DR approaches [129] project the components in KGs into low-dimensional continuous space to preserve the structural information while simplifying the manipulation. The typical steps of DR approaches include 1.) representing entities and predicates with initial embedding values; 2.) defining a scoring function that measures the plausibility scores for given facts, and 3.) learning the embeddings by maximizing the total plausibility score of observed facts. DR models can be divided into translational distance models and semantic matching ones depending on the scoring functions. Translational distance models use distance-based scoring functions, whereas the scoring functions of semantic matching models are based on similarities between entities. One of the most representative translational distance models is TransE [13]. For a given entity or predicate i, we denote by $\varepsilon(i)$ the embedding of i. Consider we are given a fact $Speak(Bill\ Gates, English)$, TransE treats the embedding of predicate $Speak$ as a translation vector so that $\varepsilon(Bill\ Gates) + \varepsilon(Speak) \approx \varepsilon(English)$ holds in the embedding space. After training, if $\varepsilon(Melinda\ Gates) + \varepsilon(Speak) \approx \varepsilon(English)$ also holds, the new fact $Speak(Melinda\ Gates, English)$ can be inferred to complete the KG. RESCAL [88] is one of the most popular semantic matching models where each predicate is associated with a matrix that models pairwise interaction between entities to capture the latent semantics of KGs. DR works suffer from two main drawbacks. First, most of the works can only perform transductive inference where when the prediction involves a previously unseen entity, the whole system needs to be re-trained to generate an embedding for the new entity. This inability to infer inductively limits the usefulness of DR systems given the volatile nature of real-world data. Second, the semantic mappings of symbols to interpretable concepts have lost during the process of projecting KG components onto a continuous space, thus the learned model is not explainable.

GNNs [133] have become a hot topic in data mining and machine learning communities in recent years. Though both DR and GNN methods utilize latent representations for modeling, the embeddings produced by DR systems are often streamlined to aid subsequent graph analytics tasks including clustering and classification while GNNs are end-to-end models designed for specific tasks. In comparison to images, graphs are irregular because nodes have different numbers of neighbors and it is impossible to decide the topology of neighbors. Therefore, traditional neural network layers, such as convolution, can not be conveniently applied to graphs. To address such problem, early GNN works [40, 106, 44] propose an architecture where the representation of a node is learned by propagating and aggregating neighbor information in an iterative manner until a stable status is reached. Recent GNN models, including R-GCN [107] and GraphSAGE [49], have demonstrated competitive performance on link prediction tasks compared to DR methods while being able to make inductive inferences. However, even with recent attempts [139] through the identification of sub-graph patterns with high relevance scores to provide limited insights in model interpretability, GNN models are, in general, not explainable.

Rule learning methods [140, 39, 90, 75] that generate first-order logic rules based on ontological and relational information presented in KGs have gained their popularity by being inductive, interpretable and transferable. Rule learners are often categorized as a type of the Graph Feature Models where high-level regularities or explicit patterns are extracted as features from the observed facts [87]. As rule learning is a symbolic method, it takes as input a KG and produces a symbolic hypothesis composed of logical rules, the learned model is fully interpretable and can be used to support prediction results for expert verification. For instance, a rule:

$$Nationality(X,Y) \leftarrow Found(X,V_0), Headquarter(V_0,V_1), City(V_1,Y)$$

states that if a person X founds a company V_0 and V_0 is headquartered at V_1 which is a city in a country Y, then X has the nationality of Y, where X, V_0, V_1 and Y are variables. This rule is abstracted from the observation in Figure 2.1 where *Bill Gates*, who has the nationality of the *US*, founded *Microsoft* at *Redmond* in the *US*. Because the variables in rules are universally quantified, rules can be used to infer new facts with previously unseen entities and even shared between the KGs in similar domains [151]. Compared to DR and GNN methods that employ numerical optimization for learning, rule learning systems that explore discrete rule space in order to retrieve high-quality rules are less robust to noise and less effective in generalizing the underlying data distribution. Nevertheless, recent works demonstrate that well-designed rule learning systems [75, 74] perform very competitively on link prediction tasks compared to other methods, and the idea of incorporating logical rules as additional knowledge to augment DR and GNN models has proven successful in improving system performance [64]. Therefore, further advancing the front of rule learning systems has a potentially significant impact on the data mining and machine learning community.

2.2 Logical Rule Learning Systems

In this section, we analyze the characteristics of existing rule learning systems by investigating three mechanically important components, including language bias, problem definition and learning paradigm. For the completeness of discussion, our focus is not limited to learning rules only from KGs. Formalism and conventions employed by systems mining rules from more general data models, including Knowledge Bases (KBs), are also included [31]. First and foremost, we introduce a set of notations that are used in this chapter and simplified from clausal logic [25]. In the formalism of definite Horn clause [89], a rule is written as:

$$L_h \leftarrow L_1, ..., \neg L_i, ..., L_n$$

where $\leftarrow$ expresses logical consequence, and L_h is the head atom and the rest are body atoms. An atom can be represented in the form of $r(t_0, ..., t_n)$ where t_j is an argument of the predicate r. The arguments in predicates are also known as terms that can be variables, constants or functions. A definite Horn clause has exactly one head atom, and if not specifically stated, all rules that are discussed in this thesis follow the convention of definite Horn clauses. We use upper-case letters to express variables and lower-case letters for constants. A function, denoted by $f(t)$, maps a term to a set of terms based on a specific pattern. Consider we are given a rule that contains only binary atoms as:

$$r_t(X,Y) \leftarrow r_1(X,V_0), ..., r_n(V_{n-1},V_n)$$

and an *instantiation* or a *grounding* of it, that is all variables in the rule are replaced with constants by instantiating each of the atoms over a background knowledge, is:

$$r_t(e_0,e_1) \leftarrow r_1(e_1,e_2), ..., r_n(e_{n-1},e_n)$$

we call (e_0,e_1) the head grounding of the rule and $(e_1,...,e_n)$ the body grounding. We denote by H_p a set containing all possible head groundings of a rule p, and define the coverage of p for a given set of pairs of constants E as:

$$C_p(E) = \{(x,y) | (x,y) \in (H_p \cap E)\} \tag{2.1}$$

and correspondingly, the coverage of a rule set P is:

$$C_P(E) = \bigcup_{p \in P} C_p(E) \tag{2.2}$$

Formally, given a target predicate r, a background knowledge B and sets of positive and negative instances of r, denoted by I^+ and I^- respectively, a rule learning algorithm aims to produce a set of rules P. When rules in P are grounded over B, $C_P(I^+)$ and $C_P(I^-)$ are satisfactory to specific criteria. The learned model is thus the rule set P.

2.2.1 Language Bias

One factor that drastically differentiates one rule learning system from another in terms of scalability and expressivity is the types of rules that a system can produce, which is governed by language bias. Language bias imposes constraints on the syntax and semantics of the language employed to express the learned rules. In other words, the rule space of a rule learner only contains well-formed rules according to a given language bias. In early Inductive Logic Programming (ILP) works where knowledge is represented as KBs that allow predicates with arbitrary arities [81], mode declaration [30] is widely used to specify language bias. For instance, to produce the rule:

$$p_0 : Locate(X,Y) \leftarrow Headquarter(X,V_0), city(V_0,Y)$$

which states that if a company X is headquartered at a city V_0 that is in a country Y, then X is located in Y, we need to manually declare all the predicates that appeared in the rule before learning. In addition, the specifications of which variables are input variables that must be instantiated before predicate grounding and which ones are output variables instantiated at the moment of predicate grounding are required. If not constrained in the language, the size of rule space will grow exponentially with the inclusion of predicates with large arities. For instance, given two predicates $r_0(V_0,V_1,V_2)$ and $r_1(V_3,V_4,V_5)$, to form a rule, r_0 and r_1 can be connected in many ways, e.g., $r_0(V_0,V_1,V_2), r_1(V_0,V_4,V_5)$ or $r_0(V_0,V_1,V_2), r_1(V_1,V_4,V_5)$ and so on. Therefore, it is considered necessary in early ILP systems to specify argument types in language biases and by enforcing type matching to restrict the rule space such that the learning procedures can be executed in a reasonable time.

In comparison to ILP, recent rule learning works [103] focus more on extracting logical rules from KGs where the implicit syntactic constraint is that only binary predicates are allowed. The adoption of the use of only binary predicates for knowledge representation is because a predicate with large arity can often be broken into a set of interrelated binary predicates. For instance, a tabular data $\{name : Alice; age : 18; gender : female\}$ can be converted into the fact $Entity(Alice, 18, female)$ where the property names are encoded as entity types. This ternary fact can be expressed by binary facts $Age(Alice, 18)$ and $Gender(Alice, female)$ without the loss of information given that the meta-knowledge is well-managed. Seemingly, the conversion from a predicate with large arity to a set of binary predicates introduces redundancies to data storage, but in well-designed graph databases, this overhead is often trivial [43]. Although the restriction to only allow binary predicates makes the mining of rules from KGs easier than from general KBs, language bias which dictates what types of rules to include is still needed to achieve better scalability. The systems proposed in this thesis take the conventions from recent works and work only with binary predicates.

It is common that a rule learning system produces various types of rules such that different rule types complement one another in the sense of expressing concepts of different abstraction levels. We here provide a discussion about the major or unique rule types that are proposed by popular rule learning systems. Most rule learning systems, including QuickFOIL [140] and ScaLeKB [21], mine *positive* rules for the deduction of facts that are unknown to the given KGs. The rule:

$$Speak(X,Y) \leftarrow Nationality(X,V_0), Language(V_0,V_1)$$

is a positive rule where new facts can be inferred through the grounding of the universally quantified variables in the rule [25]. For example, the fact $Speak(Boris\ Johnson, English)$ can be inferred with a confidence score through the grounding of the rule based on the triples $Nationality(Boris\ Johnson, UK)$ and $Language(UK, English)$. This rule is also *abstract* in that it contains no constants. Correspondingly, AMIE+ [39] and AnyBURL [75] propose to mine *instantiated* rules that contain constants to enrich the expressivity of the learned rule space. An instantiated version of rule p_0 can be:

$$Locate(X,Y) \leftarrow Headquarter(X,Redmond), city(Redmond,Y)$$

or:

$$Locate(X,Y) \leftarrow Headquarter(X,V_0), city(V_0,US)$$

where *Redmond* and US are constants. Many discussions about what types of instantiated rules to learn are provided by recently published works [39, 76, 91, 75]. In general, it has been proven that the inclusion of instantiated rules helps improve systems performance. However, it also exposes rule learners to a much larger rule space compared to learners that only include abstract rules due to the size of instantiated rules proportional to that of entities. The development of methods optimizing the mining of instantiated rules and the study of the characteristics of instantiated rules are not well researched at the moment.

The monotonicity of logical rules describes the pattern where with increasing rule complexity, e.g., addition of atoms or instantiation of variables, the coverage of the rule can either stay the same or decrease. Rules that have no negative body atoms conform to the monotonicity and are known as *monotonic* rules [37, 54]. *Non-monotonic* rules that contain negative body atoms are proven useful in expressing the inverse concepts of known predicates. For instance, the statement "married people live in the same place unless one is jailed" can be expressed by the non-monotonic rule:

$$LiveIn(X,Y) \leftarrow Marry(X,V_0), LiveIn(V_0,Y), \neg Status(Y,Jailed)$$

RuLES [54] is a system that mines non-monotonic rules for exception handling. Moreover, through the modelling of counter-examples, RuDiK [2] proposes to produce *negative* rules for the identification of contradictions in the data. By allowing the head atom to be negative, negative rules describe logical inconsistencies and thus identify erroneous triples. A negative rule:

$$\neg Nationality(X,Y) \leftarrow Nationality(X,China), Y \neq China$$

states that if a person X has the nationality of China, then X can not have other nationalities. In addition to form rules with existing predicates, RuDiK also augments the expressivity of rule spaces by automatically constructing the comparison relationships between the literals in KGs. For instance, a rule:

$$BigBrother(X,Y) \leftarrow Brother(X,Y), DOB(X,e_0), DOB(Y,e_1), e_0 > e_1$$

contains the operator $>$ to compare between ages e_0 and e_1 with DOB standing for "Date of Birth". The choice of language biases has significant impacts on both the downstream tasks a learning system aims to tackle and the system's performance in terms of scalability, predictive power and interpretability. Therefore, the design of rule learning systems requires one to first pick the right language bias for the given task and then implement other components to fulfil the language bias efficiently.

2.2.2 Problem Definition

Problem definition specifies the success conditions required to terminate the learning procedure. Rule learning is mostly formulated as a discovery problem where a learning system aims to produce a solution rule set that is satisfactory to some criteria. Various stopping criteria have been employed in existing works to handle certain tasks. Classic ILP systems, such as FOIL [99] and Aleph [116], find a solution set P' that is a subset of the rule space and satisfies the following criterion:

$$C_{P'}(I^+) = I^+ \wedge C_{P'}(I^-) = \emptyset \tag{2.3}$$

where I^+ and I^- are positive and negative instances of a target predicate, respectively. When applied to a complete and noise-free KG, a rule learning system that employs Equation 2.3 as its problem definition will end up with a perfect solution set that covers all of the positive instances and none of the negative ones. We call this problem definition an *exact* approach as it has zero tolerance to the coverage of negative instances. However, as reasoned in previous sections, real-world KGs are inherently incomplete and noisy. Exact systems often perform poorly at retrieving informative rules in practice. To mitigate this issue, *probabilistic* or *soft* approaches have been proposed where a solution set that covers negative instances is acceptable. It is also important to notice that Equation 2.3 acts as a *global* criterion, which indicates it is applied to the solution set P'. *Local* criteria that put constraints on the quality of individual rules are not compulsory in early ILP systems. Systems with a global criterion usually aim to produce a *concise* solution set that includes a relatively small number of rules and is optimized to satisfy the global criterion. In contrast, a *comprehensive* solution set contains all the qualified rules under certain local criteria. We denote by $\phi(P')$ an indicator function that returns true if for any $p \in P'$, p is considered satisfactory to a collection of pre-defined local criteria, e.g., the coverage of positive instances or prediction confidence. Popular KG rule mining systems, including AMIE+ [39], ScaLeKB [21] and AnyBURL [75], all employ a probabilistic and comprehensive problem definition that can be formally defined as:

$$\underset{P'}{\operatorname{argmax}}(|P'|;\phi(P')) \tag{2.4}$$

which is to extract all qualified rules. In general, a comprehensive solution set provides better predictive performance and interpretability than a concise set at the expense of model size and management overhead. The improvements in performance are mainly attributed to the fact that the rules in a comprehensive set can provide a more complete knowledge about target concepts. It is also common to employ both global and local criteria to form a problem definition. RuDiK [91] takes a concise and probabilistic approach where the solution set P' is constructed to optimize a weighted loss function $w(P')$ which considers the impacts of the coverage of both positive and negative instances. Its problem definition can be described as:

$$\underset{P'}{\operatorname{argmin}}(w(P');C_{P'}(I^+) = I^+ \wedge \phi(P')) \tag{2.5}$$

which indicates that RuDiK aims to find a concise set that satisfies the global conditions with qualified rules.

2.2.3 Learning Paradigm

The Learning paradigm decides the procedural details on how rules are generated. There are three types of learning paradigm in existing works. The top-down paradigm starts the search from an empty rule covering all instances. By iteratively performing refinements to specialize the empty rule and the rules derived from the empty rule, a rule set is populated with the specialized rules. For instance, AMIE+ [39] utilizes refinement operators to extend rules with: 1.) dangling atoms that introduce a new variable, 2.) instantiated atoms that contain constants, and 3.) closing atoms that share all variables with the rule to make the rule cyclic. As the proposition of new rules with the top-down paradigm often does not require background knowledge and training instances, it is prone to generate groundless rules that cover none of the instances and need to be grounded to be identified. On large-scale KGs, the grounding procedure is expensive to execute. Groundless rules that invoke the grounding procedure while not contributing to inference often introduce non-negligible overheads.

The bottom-up paradigm populates a rule set by generalizing the concrete paths extracted from KGs. For instance in Figure 2.1, given the target predicate $Nationality(X,Y)$, the path:

$$US \xleftarrow{Nationality} Bill\ Gates \xrightarrow{Found} Microsoft \xrightarrow{Locate} US$$

can be abstracted into the rule:

$$Nationality(X,Y) \leftarrow Found(X,V_0), Locate(V_0,Y)$$

by treating the originating triple $Nationality(Bill\ Gates, US)$, which is a positive instance of the target predicate, as the head atom and replacing all the constants with variables. In such a way, the generated rules are guaranteed to cover at least one instance. Bottom-up systems, including RuDIK [91], RuleN [76] and AnyBURL [75], are not required to know the metagraph or schema of a KG beforehand to generate rules. Instead, they chart the metagraph while producing rules. Because bottom-up systems sample paths by initiating graph traversal from instances, which can be achieved by only loading the neighbourhood of the originating instances into memory, it avoids the memory-hungry practice employed by top-down methods where the entire KG is often required to be pre-loaded into memory. Therefore, many memory-efficient bottom-up systems that can run on commodity computers have been proposed [90, 91].

Some of the traditional ILP systems, such as Progol [80], Aleph [116] and CILP++ [36], utilize a top-down-bottom-up strategy where an empty rule is first specialized into a bottom clause which is a compact representation of a positive instance and then the rule space is populated by generalizing the bottom clause. For instance in Figure 2.1, a bottom clause for the instance $Nationality(Bill\ Gates, US)$ can be:

$$\begin{aligned} Nationality(Bill\ Gates, US) \leftarrow & Found(Bill\ Gates, Microsoft), \\ & Marry(Bill\ Gates, Melinda\ Gates), \\ & ..., City(Redmond, US) \end{aligned}$$

and by repeatedly abstracting constants and discarding atoms, a set of specialized rules can be generated. This paradigm also guarantees that all mined rules are non-groundless in that the bottom clause is created by consulting background knowledge. Interestingly, the top-down-bottom-up paradigm can be seen as a complex version of the bottom-up strategy in that they both aim to extract rules by initiating a search around positive instances, one instance at a time.

Chapter 3

Graph Path Feature Learning

As shown in recent works [76, 75], systems mining instantiated logical rules demonstrate stronger predictive power and interpretability than ones that only include abstract rules. Motivated by this observation, we in this chapter introduce the GPFL (Graph Path Feature Learning) system which is a probabilistic logical rule learner optimized to mine instantiated rules from Knowledge Graphs (KGs). GPFL utilizes a novel two-stage rule generation mechanism with a higher-order function to control the progress of learning. Unlike existing works that ground every mined instantiated rule for evaluation, GPFL shares groundings between structurally similar rules for collective evaluation, thus is more efficient than existing works. Moreover, we investigate the presence of overfitting rules, their impact on predictive performance, and the effectiveness of a simple validation method filtering out overfitting rules. Through extensive experiments on public benchmark datasets, we show that GPFL is much more efficient than existing works in mining instantiated rules, and the prevalence of overfitting instantiated rules is too severe to be ignored by instantiated rule learners, thus must be suppressed properly.

3.1 Introduction

We consider the rule space of a rule learner as a set containing all possible rules that can be produced by the learner under various constraints. The balance between rule space complexity for model expressivity and scalability for system practicability remains one of the core challenges in rule learning. The complexity of a rule space is determined by the choice of language bias [25], which dictates what types of rules to learn. As reasoned in Chapter 2, different language biases often render rule spaces differing drastically in size. Let us take the evolution of Path Ranking Algorithms (PRAs) as an example. PRA [66] uses random walkers to extract a specific type of rules, the Closed Abstract Rules (CARs) that are cyclic sequences of predicates connecting entity pairs. For instance in Figure 3.1, given predicate $Capital\_of(X,Y)$ as the learning target, and entity pair $(Beijing, China)$ as a positive instance, we can manually induce the CAR:

$$p_0 : Capital\_of(X,Y) \leftarrow City\_in(X,Y)$$

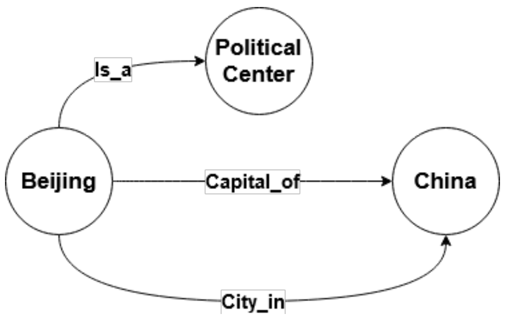


Fig. 3.1 A small knowledge graph.

which translates as: if a city X is in a country Y, then X is the capital of Y. It is apparently too general to describe the idea about capital. To explain ideas in more detail, Cor-PRA [65] proposes to include Tail Anchored Rules (TARs), a type of instantiated rule with the last variable being substituted by a constant. Again in Figure 3.1, we can induce the TAR:

$$p_1 : Capital\_of(X,Y) \leftarrow Is\_a(X,PoliticalCenter)$$

which contains the constant "Political Center". The conjunction of p_0 and p_1 can be translated as: if a city X is in a country Y, and X is a political center, then X is the capital of Y. Although still too general to describe the concept about the capital city in large countries, it is precise enough to describe the defining characteristics about the capital city in the small countries that make up the majority of countries in the world. The inclusion of instantiated rules comes with improved expressivity and often predictive power. However, it also exposes the system to a much larger rule space than only including the abstract rules that are not subject to the constants in the KG. For instance, assume the cardinality of V_1 in atom $Is\_a(X,V_1)$ is m, from a simple abstract rule:

$$p_2 : Capital\_of(X,Y) \leftarrow Is\_a(X,V_1)$$

that subsumes TAR p_1 with respect to generality, we can derive m TARs by replacing variable V_1 with constants, which is a m times growth in rule space size from one abstract rule. On large KGs with millions of relationships and entities, the scale of their rule space with instantiated rules included makes most of the existing rule learning strategies infeasible. Therefore, optimization and approximation approaches for efficient rule generation and evaluation are needed for scaling up instantiated rule miners.

3.1.1 Rule Generation

The rule generation procedure dictates how the rule space is traversed and when to stop the exploration. In other words, it is decided by the choices of problem definition and learning paradigm. Many existing works [66, 39, 76] that take a comprehensive problem definition explore the entirety or randomly sampled sub-spaces of the rule space. As the rule space that includes instantiated rules is often enormous, it is either too expensive or infeasible to search the entire rule space for high-quality rules. Besides, instantiated rules mined from randomly

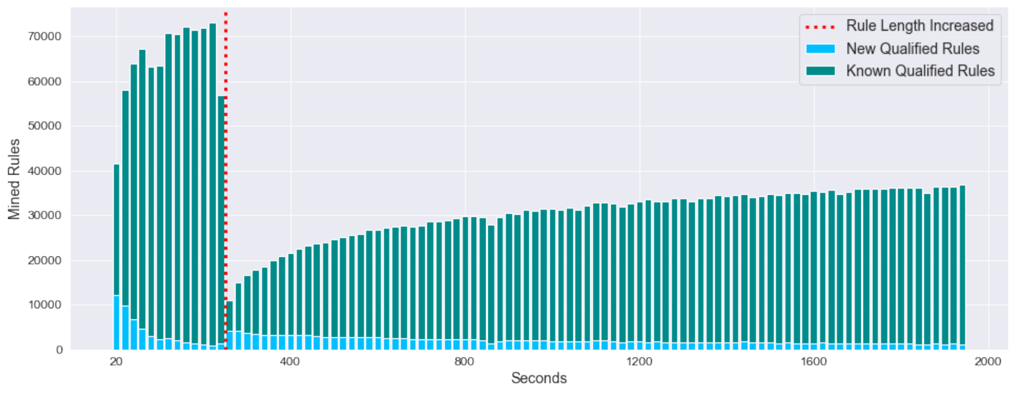


Fig. 3.2 Rules mined by AnyBURL [75] on WN18RR [121] over a period of time. Each bar represents the composition of rules mined by AnyBURL per 20 seconds, where a rule can either be known or new.

sampled sub-spaces are often subject to the locality issue. AnyBURL [75] proposes the use of a higher-order function to control the progress of rule generation. Specifically, rules of length n are mined in batches where the rules learned in previous batches are considered as known rules, and if the proportion of known rules in the current batch is above a saturation threshold, the system either progress to mine rules of length $n+1$ or terminates. Although the use of rule saturation for automatically extending the search space is desirable in that the search is exposed to the entire rule space to mitigate sampling bias, and the search early-stops when enough frequent regularities are extracted, it is inefficient at generating instantiated rules because to reach the saturation for progress, it needs to repeatedly visit the same set of frequent rules until a few less frequent yet informative rules are discovered. For instance, in Figure 3.2, the ratio between the known and new rules is relatively reasonable at the beginning of the search or after the trigger of a progression. However, the ratio drops significantly as the system runs where over 90% of the newly discovered rules are known. Therefore, the system has trouble progressing to learn long rules. This inefficiency is caused by the overwhelmingly large rule space over which the saturation is measured meaning that it is hard to converge.

3.1.2 Rule Evaluation

The rule evaluation procedure decides what rule quality measure to use and how to plan rule evaluation executions. Most of the existing works employ statistical measures such as confidence and support [39] to indicate rule quality. Statistical measures are costly to compute in that they require systems to ground rules using a backward chaining algorithm that is exponentially complex. Despite the inefficiency, most existing works ground every mined rule individually for evaluation, which leads to the main scalability bottleneck. Recent works propose incorporating pre-trained embeddings into the measure of rule quality to mitigate the scalability issue. For instance, RLvLR [90] uses quality measures based on embedding similarities to score rules. The disadvantage of embedding-augmented methods is that the training of embedding models on large KGs itself is often not trivial.

3.1.3 Approach

In this chapter, we propose the GPFL system, a novel probabilistic rule learner optimized to mine instantiated rules. We use templates that are acyclic abstract rules to optimize both rule generation and evaluation. Specifically, GPFL utilizes a two-stage rule generation mechanism. In the generalization stage, GPFL optimizes AnyBURL's higher-order function by saturating the template space that contains all possible templates to create a set of frequent templates. As the template space is usually smaller in size by orders of magnitude than the entire rule space over which AnyBURL measures its saturation, template saturation is much easier to converge. Thus the system can learn long rules with ease. Inspired by the idea of Query Pack [8], in the specialization stage, GPFL makes optimized use of template groundings for both deriving and evaluating instantiated rules. In particular, as instantiated rules derived from the same templates are structurally similar, instead of grounding every mined rule individually, GPFL only grounds templates and uses the templates' groundings to evaluate the derived instantiated rules collectively. In such a way, GPFL significantly reduces the number of the invocations of expensive grounding procedures than in existing works. These optimizations allow GPFL to learn instantiated rules much more efficiently than existing approaches in terms of quality and quantity. Moreover, as instantiated rules are more specialized than abstract ones, they are more likely to overfit the training set. GPFL removes the overfitting rules via a simple validation method to further improve predictive performance. Our contributions can be summarized as follows:

- We propose a novel probabilistic rule learner optimized to mine instantiated rules via the use of a two-stage rule generation mechanism.
- To the best of our knowledge, this is the first work that studies the overfitting issue of instantiated logical rules in depth.
- Through extensive experiments on public benchmark datasets, we observe that GPFL: significantly reduces the runtime on evaluating instantiated rules; is much more efficient than AnyBURL at mining instantiated rules; improves the predictive performance of learned rules by removing overfitting rules via validation and shows competitive performance on knowledge graph completion task in comparison to state-of-the-art baselines.

3.2 Methodology

In this section, we introduce in detail how the proposed system works. First, we define a set of notations that will be used throughout this thesis. Second, we specify the language bias employed in this work. Finally, after giving an overview about the design of the system, we elaborate on the system's core components, including the two-stage rule generation mechanism and the collective rule evaluation strategy.

3.2.1 Preliminaries

We denote by $\mathscr{G} = (\mathscr{E}, \mathscr{R}, \mathscr{T})$ a KG. A ground atom in $\mathscr{G}$ is in the form of a triple $r(e_i, e_j) \in \mathscr{T}$ where $r \in \mathscr{R}$ is a predicate representing concepts in the domain, and $e_i, e_j \in \mathscr{E}$ are entities also known as constants in logic terms. For a ground atom $r(e_i, e_j)$, e_i is the subject of the triple and e_j the object. We denote by Sub_r and Obj_r the sets that contain all the subjects and objects of the ground atoms with predicate r in a KG, respectively. A non-ground logical rule, that contains at least one variable, is the following:

$$p_3 : r_t(X,Y) \leftarrow r_1(X,V_0), ..., r_n(V_n,Y)$$

where $r_t(X,Y)$ is the head atom with a target predicate r_t and target variables X and Y, and the rest are body atoms. We use lower-case letters for constants and upper-case letters for variables. The length of a rule is the count of body atoms. Rule p_3 states that if the conditions in the body can be grounded in the KG, then the head atom can be inferred as a fact. A path is a sequence of ground atoms extracted by traversing a KG. When a non-ground rule is instantiated into a path, that is all atoms in the rule are grounded, the path is known as an instance of the rule. For example, the path:

$$p_4 : r_t(e_0,e_1) \leftarrow r_1(e_0,e_2), ..., r_n(e_n,e_1)$$

is an instance of p_3 and this instantiation relationship between p_3 and p_4 is denoted by $p_4 \in In(p_3)$ where $In(p)$ returns a set containing all of the instantiation of a rule p over a KG. The pair (e_0, e_1) that instantiates the target variables in the head of p_3 is called a head grounding and is denoted by h_{p_3}, and the ordered sequence $b_{p_3} = (r_1(e_0,e_2), ..., r_n(e_n,e_1))$ is a body grounding, that explains the existence of $r_t(e_0,e_1)$. For *closed* rules where all of the target variables also occur in body atoms, head groundings can be generated by simply grounding the rule body. However, for *open* rules where only a subset of target variables occur in the body, head groundings are generated through the Cartesian product of the groundings of the target variable instantiated in the body and that of the target variable in the head. For instance, given an open rule:

$$p_5 : r_t(X,Y) \leftarrow r_1(X,V_0), ..., r_n(V_n,V_{n+1})$$

a body grounding that includes $r_1(e_0,e_1)$, and $Obj_{r_t} = \{e_2, e_3\}$, the head groundings of p_5 thus are $\{(e_0,e_2),(e_0,e_3)\}$. Given a non-ground rule p, we define a set containing all the head groundings of p over a KG as:

$$H_p = \{(x,y) | (x,y) = h_{p'}, p' \in In(p)\} \tag{3.1}$$

In this work, we aim to mine *probabilistic* first-order logic rules. Specifically, a probabilistic rule is one that does not hold exactly and is assigned a confidence score to reflect the probability of it being true for a given KG. For instance, a probabilistic rule is:

$$r_t(X,Y) \leftarrow r_1(X,V_0),...,r_n(V_n,V_{n+1}) \quad [0.6]$$

where [0.6] indicates that the facts covered by the rule have a probability of 60% to be true.

3.2.2 Language Bias

As introduced in Chapter 2, Language bias, as a prior knowledge along with semantic bias, is used extensively in rule learning systems to restrict rule space by specifying the desired types of rules to include [25]. For rule learners that generate rules based on paths extracted from KGs, the implicit syntactic restrictions are that only binary atoms are allowed, and adjacent atoms share the same variables or constants. In this work, we only consider *straight* rules where a variable or constant can occur at most twice in the body atoms to avoid cycles. Also, we do not generate trivial rules that self-loop, such as:

$$r_t(X,Y) \leftarrow r_1(X,V_0),...,r_n(V_n,X)$$

Now, we introduce some of the terms used throughout this work. Given a rule:

$$r_t(X,Y) \leftarrow r_1(X,V_0),r_2(V_0,V_1),...,r_n(V_n,V_{n+1})$$

we call the variable X the original variable in that the body atoms are originated from it; the variable Y the free variable; variables such as V_0 the connecting variables in that they connect adjacent atoms, and the non-connecting variable V_{n+1} in the last body atom the tail variable. In this terminology, a rule is *closed* if the free variable Y is also the tail variable, and is *open* if the free variable does not occur in the body atoms.

In this work, we use the following types of rules to make up our language bias:

$$\begin{aligned}
\textbf{Template}: & r_t(X,Y) \leftarrow r_1(X,V_0),...,r_n(V_n,V_{n+1}) \\
\textbf{HAR}: & r_t(X,e_k) \leftarrow r_1(X,V_0),...,r_n(V_n,V_{n+1}) \\
\textbf{BAR}: & r_t(X,e_i) \leftarrow r_1(X,V_0),...,r_n(V_n,e_j) \\
\textbf{CAR}: & r_t(X,Y) \leftarrow r_1(X,V_0),...,r_n(V_n,Y)
\end{aligned}$$

where a template is an open abstract rule; a Head Anchored Rule (HAR) is a specialization of a template where the free variable is substituted with a constant; a Both Anchored Rule (BAR) is a specialization of a HAR where the tail variable is replaced with a constant, and a CAR is a closed abstract rule. Collectively, abstract rules include CARs and templates, and instantiated rules include HARs and BARs. In particular, templates are used as intermediate rules for generating HARs and BARs only, and will not be included in the learned rule set for inference. This is because templates as rules are too general to differentiate predictions. A HAR characterizes potential candidates that share relationships of type r_t with an entity e_k by a pattern, whereas a BAR highlights a pattern involving the co-occurrence between entities e_i and e_j. The CAR is a base rule type included in language biases employed by most of the existing works in that it is often small in size but provides a good base predictive performance.

For instance, consider we have a template:

$$Speak(X,Y) \leftarrow BornIn(X,V_0), Country(V_0,V_1)$$

which states that if a person X is born in a city V_0 of a country V_1, then X speaks language Y, it is noticeable that the free variable Y is not connected to any of the body atoms. When this template is grounded, the variable Y can be instantiated to any valid entities in the target KG, and the instantiation of it is irrelevant to the grounding choice of the body atoms. Therefore, template as a type of rule is too general to describe insightful regularities. By anchoring the free variable to an entity, a HAR that is possibly useful can be created. One possible instantiation is:

$$Speak(X,English) \leftarrow BornIn(X,V_0), Country(V_0,V_1) \quad [0.8]$$

which states that there is a probability of 80% that a person who involves in the *BornIn* and *Country* facts in the KG speaks *English*. Generally, the pattern represented by this HAR is false as the language a person speaks is often related to the person's birthplace or where the person lives. However, when the target KG is biased or incomplete where most persons included in the KG speaks *English*, the discovery of this HAR can be used to make accurate predictions in the context of the KG or reveal the bias to the database administrator. By further instantiating the tail variable, a BAR that specifies co-occurrence can be constructed. A BAR based on the previous HAR can be:

$$Speak(X,English) \leftarrow BornIn(X,V_0), Country(V_0,UK) \quad [0.95]$$

which states that it is highly possible that a person who is born in a city of the *UK* speaks *English*. BARs with high confidence score can provide interesting insights between pairs of entities, which is not possible with abstract rules. CARs represent alternative explanations of a concept. For instance, a CAR:

$$Speak(X,Y) \leftarrow BornIn(X,V_0), Country(V_0,V_1), Language(V_1,Y) \quad [0.9]$$

states that if a person X is born in a city V_0 of a country V_1 in which the language Y is spoken, then it is highly possible that the person X speaks the language Y. The pattern expressed by this CAR can be seen as an explanation of the concept *Speak*. The inclusion of constants makes the patterns represented by the instantiated rules, including HARs and BARs, much more specific than that represented by abstract rules, which in turn implies that the instance coverage of instantiated rules is generally less broad than that of abstract rules. With small coverage, it seems that instantiated rules are more precise than abstract ones, and the exact approach can be employed. However, due to the inherently incomplete and noisy nature of KGs, it is still necessary to measure instantiated rules in a probabilistic manner to counter the existence of false positive and negative triples.

The reason for selecting these rule types is based on the assumptions of concept stratification and deconstruction. Concept stratification assumes that learning targets usually have different explanatory complexities. Thus rule types of different complexities should be included in the rule space to adapt to different targets. For instance, given a correct prediction $r_t(e_0, e_1)$ and an incorrect one $r_t(e_2, e_1)$, both are suggested by the HAR:

$$r_t(X, e_1) \leftarrow r_1(X, V_0) \quad [\alpha_1]$$

As they are suggested with the same confidence α_1, the system can not distinguish one from another. This is a case where the rule space is too general for the learning target. When we allow the more specific BAR in the rule space, and we know that the BAR:

$$r_t(X, e_1) \leftarrow r_1(X, e_3) \quad [\alpha_2]$$

predicts $r_t(e_0, e_1)$ with confidence α_2, the system then can treat the predictions differently. The inclusion of both HAR and BAR is an attempt to stratify the concepts that can be expressed by the system for better adaptivity.

Concept deconstruction assumes a complex concept can be expressed by the combination of simple concepts. For instance, a complex rule that has more than one constant in its body atoms is as follow:

$$r_t(X_1, e) \leftarrow r_1(X_1, e_0), r_2(e_0, e_1)$$

and it can be expressed by the conjunction of BARs:

$$\begin{aligned} r_t(X_2, e) &\leftarrow r_1(X_2, e_0) \\ r_t(X_3, e) &\leftarrow r_1(X_3, V_0), r_2(V_0, e_1) \end{aligned}$$

in that X_1 has the same domain as $X_2 \cap X_3$. The meaning of the concept deconstruction assumption is that: as complex concepts concisely expressed by more specific rules can be described through the conjunctions of less specific rules, we do not need to invent new learning frameworks for the mining of more specific rules, instead, it is enough for us to learn a comprehensive set of less specific rules under the existing learning frameworks.

We here provide a discussion about the instantiated rule types employed in existing works. Mining association rules from RDF databases [1, 17, 6] is a line of research that treats the subjects, predicates and objects as conditions to form a zeroth-order logic rule. For instance, an association rule:

$$\{Speak, English\} \leftarrow \{BornIn, London, Country, UK\} \quad [0.95]$$

indicates that if an entity is related to predicates *BornIn* and *Country* and entities *London* and *UK*, it is also involved in predicate *Speak* and entity *English*. In comparison to first-order logic rules, association rules are not parameterized where the explicit relations between entities and predicates in the body atoms are not encoded. Specifically, the body of an association rule is an itemset where predicates and entities are treated as same-class citizens, which introduces ambiguity in interpretation. For example, given $\{BornIn, London, Country, UK\}$, one can infer triples $BornIn(X, London)$ or $BornIn(X, UK)$. Therefore, first-order logic rules that are more expressive and stable than association rules have attracted more attentions in the KG rule mining community. RuleN [76] uses an instantiated rule type with existential quantifier in the body atom. It is formally defined as:

$$R(X, e_0) \leftarrow \exists V_0 \; R(X, V_0) \quad [\alpha]$$

and an example of it is:

$$Speak(X, English) \leftarrow \exists V_0 \; Speak(X, V_0) \quad [0.6]$$

which states that if for a person X there exists a triple $Speak(X, V_0)$, then there is a probability of 60% that X speaks *English*. This type of rule serves as a statistical function that can be replaced with global analysis. AMIE+ [39] allows the addition of instantiated atoms to closed rules. For instance, AMIE+ allows the rule:

$$Speak(X, Y) \leftarrow Gender(X, Male), BornIn(X, V_0), Country(X, V_1), Language(V_1, Y) \quad [0.8]$$

where the instantiated atom $Gender(X, Male)$ describes a property of the entity represented by X. The requirement of the rules being closed makes AMIE+'s rule space less broad and expressive than ours. RuDiK [91] allows the inclusion of constants under the condition that there is only one possible instantiation of a variable in the rule. For instance, given an abstract rule:

$$President(X, Y) \leftarrow BornIn(X, V_0), Country(X, Y)$$

RuDiK allows the instantiated version of it as:

$$President(X, US) \leftarrow BornIn(X, V_0), Country(X, US)$$

which states that over all possible groundings, only if a person X is born in the US, X can be the *President* of the US. Our choice of the language bias is largely inspired by AnyBURL [75], where HAR and BAR are employed to improve the predictive power and interpretability. In contrast to their bottom-up learning mechanism, we utilize the proposed template to enable a two-stage learning mechanism that optimizes the generation and evaluation of instantiated rules.

3.2.3 Algorithm Overview

Now we introduce the GPFL algorithm and discuss the design idea in detail. Above all, GPFL is designed to be a discriminative learner that mines a comprehensive set of rules for one target predicate at a time. For a given target predicate $r_t \in \mathscr{R}$, we denote by I^+, I_v^+ and I_t^+ the sets of positive training, validation and test instances of r_t in $\mathscr{G}$. As shown in Algorithm 3.1 which generates rules for a target r_t, GPFL starts by initializing the rule set F, and then by calling the `Generalization` procedure, a rule frequency map M is returned. Map M stores key-value pairs where the key is an abstract rule and the corresponding value is the occurrences of the rule counted during generalization. In our design, we allow the use of time and space constraints to terminate the system prematurely to accommodate tasks with diverse requirements. Therefore, it is important to sort the abstract rules in M in order to make rules that are more likely to be frequent patterns visited first in the specialization loop. The `Sort` procedure resolves this by first dividing rules into CARs and instantiated rules of different lengths, then sorting rules in each division by frequency in descending order, and eventually assembling the sorted divisions into a list P by adding CAR division first and then divisions of instantiated rules with increasing length.

For each abstract rule $p \in P$, GPFL grounds it over $\mathscr{G}$ to produce groundings G. Groundings can be used to score the abstract rule p if p is a CAR, or to derive and evaluate instantiated rules. We define a scoring procedure `Score` that first measures the quality of a rule and then decides if the rule is good enough to be included in the rule set. Various rule quality measures have been proposed in existing works. In this work, we employ three popular measures, namely the standard confidence [39], the smooth confidence [75] and the Partial Completeness Assumption (PCA) [39]. First, we define the support of a rule p as:

$$supp(p) = |H_p \cap I^+| \tag{3.2}$$

which counts the number of positive instances covered by the rule. Then, the standard confidence is defined as:

$$sc(p) = \frac{supp(p)}{|H_p|} \tag{3.3}$$

and smooth confidence as:

$$smc(p) = \frac{supp(p)}{\eta + |H_p|} \tag{3.4}$$

where η is an offset used to cope with the bias which tends to assign high confidence value to rules that only make a few predictions. In contrast to $sc(p)$ and $smc(p)$ that operate under the Closed World Assumption (CWA), PCA assumes functionality in target predicates. In particular, given a target r_t, for every e_i such that $r_t(e_i, e_j) \in \mathscr{T}$, PCA only treats triples $r_t(e_i, e_k) \notin \mathscr{T}$ with

Algorithm 3.1: Rule Generation for a Target Predicate

```
Input   : $\mathscr{G}, I^+, sat, bs, len$
Output : learned rule set $F$
1  Initialize empty set $F$;
2  $M \leftarrow$ Generalization($\mathscr{G}, I^+, sat, bs, len$);
3  $P \leftarrow$ Sort($M$);
4  for $p \in P$ do
5      $G \leftarrow$ Ground($\mathscr{G}, p$);
6      if $p$ is a CAR then
7          if Score($p, G$) then
8              $F \leftarrow F \cup p$;
9          end
10     else
11         $S \leftarrow$ Specialization($p, G, I^+$);
12         for $s \in S$ do
13             if Score($s, G$) then
14                 $F \leftarrow F \cup s$;
15             end
16         end
17     end
18     if Constraints() then
19         Break;
20     end
21 end
22 return $F$;
```

$e_k \in \mathscr{E}$ as negative examples. Formally, the negative examples of PCA is defined as:

$$I^-_{pca} = \bigcup_{x \in Sub_{r_t}} \{(x,y) | y \in \mathscr{E}, r_t(x,y) \notin \mathscr{T}\} \tag{3.5}$$

and the corresponding PCA confidence thus is:

$$pca(p) = \frac{supp(p)}{H_p \cap (I^-_{pca} \cup I^+)} \tag{3.6}$$

In addition, we define the head coverage of a rule as:

$$hc(p) = \frac{supp(p)}{|I^+|} \tag{3.7}$$

which measures the recall of p. From line 11 to 16 in Algorithm.3.1, GPFL specializes a template p with distinct constants occurred in G and I^+ into HARs and BARs, and scores each instantiated rule $s \in S$ with G which is the groundings of p instead of that of s itself. In such a way, GPFL evaluates instantiated rules collectively without the need of grounding each instantiated rule individually. At the end of the loop, the system checks whether the time and space constraints are met or not. If `Constraints` returns true, the system terminates.

Algorithm 3.2: Generalization Procedure

```
   Input  : $\mathscr{G}, I^+, sat, bs, len$
   Output : rule frequency map $M$
 1 Initialize empty set $T$ and map $M$;
 2 $c, sat' \leftarrow 0$;
 3 do
 4     $i \leftarrow$ randomly sample an instance from $I^+$;
 5     $A \leftarrow$ PathSampler($\mathscr{G}, i, len$);
 6     for $a \in A$ do
 7         $c \leftarrow c + 1$;
 8         $p \leftarrow$ Abstraction($a$);
 9         $P \leftarrow P \cup p$;
10         Update $M$ with $p$;
11         if mod($c, bs$) = 0 then
12             $sat' \leftarrow \frac{|M.keys \cap P|}{|P|}$;
13             $T \leftarrow \emptyset$;
14             if $sat' > sat$ then
15                 Break;
16             end
17         end
18     end
19 while $sat' < sat$;
20 Return $M$;
```

3.2.4 Rule Generation

Rule generation in GPFL takes two stages: the generalization phase for the discovery of abstract rules and the specialization phase for the derivation of instantiated rules. Algorithm.3.2 details the `Generalization` procedure. It takes a KG $\mathscr{G}$, positive instances I^+, a saturation threshold *sat*, a batch size *bs* and the maximum length of rules *len* as inputs, and produces the rule frequency map M. GPFL randomly samples an instance $i \in I^+$ where the entities in i are treated as starting nodes. Then, `PathSampler` is called to traverse the *len*-hop neighbourhood of i to sample a set of paths A. GPFL is designed as an in-disk system, which does not require the loading of entire KG into memory, to enable the application to large-scale KGs. Specifically, given an instance i, GPFL requests the *len*-hop neighbourhood of i from a graph database hosting the KG, loads the neighbourhood into memory and then uses random walkers to extract valid paths originated from i. In comparison to AnyBURL's approach that samples one path at a time, GPFL samples a batch of paths at a time to avoid the excessive calling of queries operated on the in-disk graph database. Each path $a \in A$ is turned into an abstract rule p by the procedure `Abstraction` in which all of the constants in a are replaced with distinct variables. The abstract rule p is then added to the current batch P and used to update map M by logging p as its key and the occurrences of p seen so far as its value. Therefore, the known rules in current batch P is the intersection of P and the rules in M. When the path counts c is a multiple of the pre-defined batch size *bs*, current

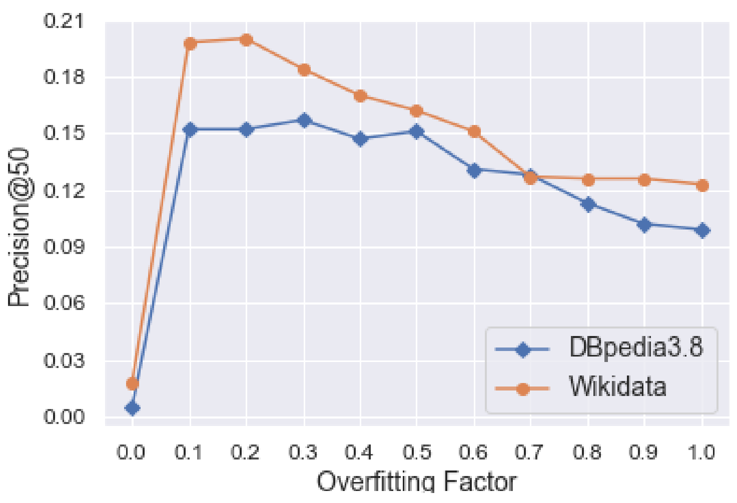


Fig. 3.3 Global average precision of top-50 rules over overfitting factors.

saturation sat' is updated to the ratio of the known rules to all rules in the current batch, and if sat' is greater than the saturation threshold sat, the `Generalization` procedure terminates.

The `Specialization` procedure in Algorithm.3.1 takes as inputs a template p, groundings G and instances I^+ to produce S which is a set of instantiated rules derived from p. Consider we are given a template:

$$p_6 : r_t(X,Y) \leftarrow r_1(Y,V_0), r_2(V_0,V_1)$$

where X is the free variable, Y the original variable, and V_1 the tail variable, and positive instances of r_t:

$$I^+ = \{(e_0,e_1),(e_0,e_2),(e_1,e_3)\}$$

we create HARs from p_6 by substituting the free variable X with constants in $Sub_{r_t} = \{e_0,e_1\}$ according to I^+. For instance, a possible HAR that can be derived from p_6 is:

$$p_7 : r_t(e_0,Y) \leftarrow r_1(Y,V_0), r_2(V_0,V_1)$$

where X is substituted with $e_0 \in Sub_{r_t}$. HARs created in this way are likely to be groundless because the groundings G are not involved in the creation process. To avoid the proposition of groundless rules, we only keep the HARs from which at least one valid BARs can be derived. A valid BAR is created by substituting the tail variable in a HAR with a constant that is linked to the constant replacing the free variable via the grounding of the original variable. Specifically, consider we have a grounding of p_6 as:

$$r_t(e_0,e_1) \leftarrow r_1(e_1,e_2), r_2(e_2,e_4)$$

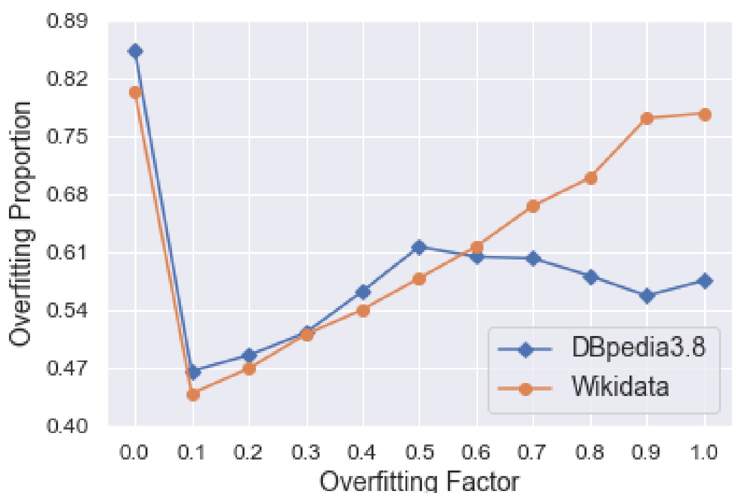


Fig. 3.4 Overfitting proportions over overfitting factors.

we denote it in compact form by (e_1, e_4) which is the pair of constants replacing the original and free variables. Given all of the groundings of p_6 in compact form as:

$$G = \{(e_1, e_4), (e_2, e_3), (e_3, e_5)\}$$

by joining pairs in G and I^+ on the constants at the position of the original variable, we can infer $\{(e_0, e_4), (e_0, e_3), (e_1, e_5)\}$ which contains pairs of constants linked by the grounding of the original variable. Therefore, a BAR:

$$p_8 : r_t(e_0, Y) \leftarrow r_1(Y, V_0), r_2(V_0, e_4)$$

can be derived from p_7 based on (e_0, e_4) and is considered valid. Therefore, p_7 is guaranteed to have at least one grounding. For HARs that have no valid BARs, we do not add them in S. When the `Specialization` procedure finishes, GPFL will start evaluating the instantiated rules in S in a collective manner.

3.2.5 Collective Rule Evaluation

We observe that the instantiated rules derived from the same templates share the same sequence of predicates. This structural similarity introduced by the deductive dependency implies that the instantiated rules derived from the same templates share either the same set or a subset of the deriving templates' groundings. For instance, given the groundings G of template p_6 and a new BAR derived from p_6:

$$p_9 : r_t(e_1, Y) \leftarrow r_1(Y, V_0), r_2(V_0, e_5)$$

we can infer that the groundings of p_7, p_8 and p_9 are subsets of G with that of p_8 being $\{(e_1, e_4)\}$ and that of p_9 being $\{(e_3, e_5)\}$. Therefore, instead of invoking the expensive grounding procedure

Data	#Entities	#Relationships	#Predicates
DBpedia3.8	2.20M	11.02M	650
Wikidata	4.00M	8.40M	430
FB15K-237	14.54K	310K	237
WN18RR	40.94K	93K	11

Table 3.1 Statistics of the benchmark datasets.

on every instantiated rule, which leads to the main scalability bottleneck for instantiated rule miners, GPFL grounds only the templates and uses the groundings of the templates to evaluate instantiated rules collectively. In such a way, we substantially reduce the runtime wasted on the invocations of grounding procedure on large KGs to achieve better efficiency.

3.3 Experiments

In this section, we establish the effectiveness of GPFL through empirical studies on four public benchmark datasets. By comparing the rules produced by GPFL and AnyBURL, we demonstrate that GPFL is much more efficient than AnyBURL at mining high-quality instantiated rules. Through evaluating rules over KGs of different sizes, we show that the collective approach implemented in GPFL significantly reduces the runtime on evaluating instantiated rules over the baseline approach adopted by many of existing works. We formally define the overfitting rules and study the characteristics and effects of overfitting rules through carefully designed experiments with GPFL. By filtering out overfitting rules with a simple validation method, improvements on predictive performance are observed with both GPFL and AnyBURL. At last, we report that GPFL performs competitively, with and without validation applied, on Knowledge Graph Completion (KGC) task in comparison to state-of-the-art logic-based and embedding-based methods.

3.3.1 Datasets

We select four publicly available benchmark datasets, including FB15K-237 [121], WN18RR [26], DBpedia3.8 [3] and the RDF version of Wikidata [39], for experimental evaluations. The statistics of these datasets are reported in Table.3.1. FB15K-237 and WN18RR are popular benchmarks for evaluation on KGC task. Both FB15K-237 and WN18RR are modified versions of the original datasets proposed in [13] to mitigate the test set leakage problem introduced by the reverse of test triples being present in the training set. DBpedia3.8 and Wikidata are large-scale KGs mostly used for testing the scalability and rule mining capability of rule learning systems.

		DBpedia3.8				Wikidata			
Quality	System	All	len=1	len=2	len=3	All	len=1	len=2	len=3
All	GPFL	3.97M	59.89K	1.36M	2.45M	563K	18.21K	264K	280K
	AnyBURL	269K	44.61K	224K	0	163K	21.92K	141K	0
High	GPFL	2.97M	22.38K	691K	2.25M	271K	8.94K	133K	129K
	AnyBURL	83.31K	13.79K	69.51K	0	55.34K	9.39K	45.95K	0
Extreme	GPFL	10.83K	715	9.57K	554	12.04K	739	5.52K	5.77K
	AnyBURL	1.46K	137	1.33K	0	710	25	685	0

Table 3.2 Sizes of learned rules grouped by length (len) and quality level.

3.3.2 Implementation

Unlike most of the existing works that are either implemented in memory or optimized on relational databases, GPFL is implemented in Java on top of the Neo4j[1] graph database. GPFL uses the core Neo4j API to traverse graph databases for path sampling and rule grounding. All experiments are conducted on AWS EC2 instances that have 8 CPU cores and 64GB RAM. GPFL allows flexible control on scalability through the adjustments of hyper-parameters to accommodate the specification of different running machines. For scalability-related parameters, we set them to values that push the running machines to limit. We have made GPFL publicly available at https://github.com/irokin/GPFL.

3.3.3 Rule Mining

In this experiment, we demonstrate that GPFL is much more efficient than AnyBURL in mining high-quality instantiated rules. We used the smooth confidence with $\eta = 5$ as the quality measure, and set confidence threshold to 0.001, support threshold to 3 and head coverage threshold to 0.001 to prune low-quality rules. We follow the standard used in [39, 90] to classify rules into high quality rules ($smc(p) \geq 0.1$ and $hc(p) \geq 0.01$), and extremely high quality rules ($smc(p) \geq 0.7$). We executed both GPFL and AnyBURL for 15000s with 6 threads. The sizes of rules grouped by different lengths and quality levels are reported in Table.3.2. GPFL learns more than 10 times the amount of rules produced by AnyBURL on DBpedia3.8, and discovers much more high-quality (High) and extremely high quality (Extreme) rules. Noticeably, AnyBURL failed to produce any rules of length 3 because it had trouble saturating rules of length 2 on both datasets, whereas GPFL succeeded in meeting the saturation threshold and proceeded to generate rules of all lengths.

3.3.4 Rule Evaluation Efficiency

In this experiment, we demonstrate the superiority of the collective strategy utilized by GPFL in evaluating instantiated rules over the baseline strategy that grounds every mined rule for evaluation. For fair comparison, we implemented the baseline approach on the Neo4j graph

[1]https://github.com/neo4j/neo4j

	WN18RR		FB15K-237		DBpedia3.8		WikiData	
Type	Baseline	GPFL	Baseline	GPFL	Baseline	GPFL	Baseline	GPFL
All	195.89	3.63	703.16	9.45	1285.62	69.41	1024.11	48.12
CAR	0.84	0.84	2.99	2.81	21.54	21.49	16.13	16.76
len=1	11.09	0.36	52.63	0.56	499.91	13.27	371.39	8.72
len=2	33.09	0.54	123.47	0.61	537.36	15.44	467.14	10.32
len=3	150.85	1.87	524.05	5.46	294.48	22.84	169.45	12.32

Table 3.3 Runtime with different rule evaluation strategies measured in various rule groups and reported in seconds.

database as well. We select WN18RR, FB15K-237, DBpedia3.8 and WikiData to account for the effect of the size of KGs on experiment outcomes. For FB15K-237, DBpedia3.8, WikiData, we randomly select 20 targets for evaluation, and for WN18RR, we evaluate all 11 relationship types. The rule set for each benchmark dataset are prepared beforehand and divided into CARs and instantiated rules of different lengths (e.g., $len = 1$) for fine-grained evaluation. To ensure experiments can be finished in a reasonable time, we allow the evaluation of each target to run for at most 30 minutes. We report the average runtime over all evaluated targets. As shown in Table.3.3, GPFL runs significantly faster than the baseline on all testing datasets when evaluating instantiated rules. For instance, it takes GPFL 1.87s to evaluate the same set of instantiated rules of length 3 that takes the baseline 150.85s to run on WN18RR. When evaluating CARs, the collective strategy in GPFL is reduced to the baseline strategy, thus we observe similar performances.

3.3.5 Overfitting Analysis

In this section, we formally define the overfitting rules, study the presence and effect of overfitting rules with GPFL, and report our observations. To understand the motivation, we first define metrics indicating the predictive performance of learned rules. The test precision of a rule p is defined as:

$$prec_t(p) = \frac{supp_t(p)}{H_p \backslash (I^+ \cup I_v^+)} \tag{3.8}$$

where $supp_t(p) = |H_p \cap I_t^+|$. We propose and use the Global Average Precision (GAP), the average of the average test precision of top-k rules of each target over all targets, as the performance indicator, where the rules are sorted by a quality measure. Formally, given a set of target predicates R, we define GAP as:

$$gap = \frac{\sum_{r_t \in R}(\frac{\sum_{p \in P_{r_t}} prec_t(p)}{|P_{r_t}|})}{|R|} \tag{3.9}$$

where P_{r_t} contains the top-k rules with the target predicate r_t. Similarly, we also measure the Global Average Quality (GAQ) over target predicates by replacing the precision function in Equation.3.9 with a quality measure function. As shown in Table.3.4, we report the GAP and

GAQ results over the different selections of top rules on DBpedia3.8 and Wikidata. Counter-intuitively, by removing certain high-confidence rules via a validation method, we observe consistent improvements on GAP on both datasets even though the GAQ drops dramatically. We argue this phenomenon is partially caused by the presence of overfitting rules. In this rest of this section, we show through the experiment results to support this argument.

Similar to overfitting models that over-perform on training set yet underperform on the test set, overfitting rules are considered high-quality when measured on training set yet have low test precision. Therefore, we consider a rule overfitting if its test precision is smaller than 10% of its quality. The choice of 10% is based on our experimental observations. To identify and remove overfitting rules, a simple solution is to measure the precision of rules on a validation set and filter out rules with a validation precision smaller than θ percent of the quality. We name θ the overfitting factor in this simple validation method and set it to 0.1 in the following experiments. We select DBpedia3.8 and Wikidata for experiments where we randomly select 20 targets from each dataset. We use the top 6000 rules from each target to create a collection of top rules and conduct an overfitting analysis on this rule collection.

We set out to answer following questions:

1. Which types of rules make up the largest portion of overfitting rule.
2. Which types of rules are more likely to be overfitting.
3. The effect of different choices of quality measure on the presence of overfitting rules.
4. The effectiveness of the validation method at removing overfitting rules.
5. The effect of the removal of overfitting rules on predictive performance.

To answer these questions, we need to define a set of terms first. We consider the overfitting rule proportion, denoted by $ORP_s(t)$, as the proportion of overfitting rules of type t to a rule space s. The domain of t includes "All" as in all rule types, CAR and instantiated rule of different lengths. The domain of s includes "all" as in the space of all learned rules, "or" as in the space of all overfitting rules, and "type" as in the space of all rules of type t. For instance in Table.3.5, with standard measure and validation on Wikidata, $ORP_{type}(len = 1)$ indicates the proportion of overfitting instantiated rules of length 1 (as t is $len = 1$) to all instantiated rules of length 1 (as s is $type$), which is 0.469. In other words, $ORP_{type}(len = 1)$ states that the probability of an instantiated rule of length 1 being overfitting is 46.9%. Similarly, we define the rule proportion, denoted by $RP_{all}(t)$, as the proportion of all rules of type t to all learned rules. Again in Table.3.5, with standard measure and validation on Wikidata, $RP_{all}(CAR)$ is 0.038, which indicates that CARs make up 3.8% of all rules. We specifically name $ORP_{all}(All)$ the overfitting proportion which measures the overfitting rules to all rules ratio.

By analyzing Table.3.5 horizontally, we can answer questions 1 and 2. We observe that the ORP_{or}, the overfitting rules of certain types to all overfitting rules ratio, of instantiated rules, are generally much greater than that of CAR. On Wikidata with smooth measure and without validation, $ORP_{or}(len = 2)$ is 0.583 and $ORP_{or}(len = 3)$ is 0.351, which means over 93% of

		DBpedia3.8		Wikidata	
Top-k	Validation	Precision	Quality	Precision	Quality
5	Yes	0.163	0.649	0.26	0.605
	No	0.023	0.914	0.05	0.918
10	Yes	0.183	0.618	0.247	0.586
	No	0.011	0.907	0.045	0.912
20	Yes	0.184	0.584	0.237	0.592
	No	0.012	0.897	0.038	0.901
50	Yes	0.152	0.536	0.198	0.553
	No	0.005	0.879	0.018	0.881
100	Yes	0.144	0.511	0.17	0.543
	No	0.004	0.859	0.015	0.855

Table 3.4 Global average precision and quality over top-k rules on DBpedia3.8 and Wikidata with and without validation applied. Smooth confidence is employed to measure rule quality and rank rules.

overfitting rules are long ($len > 1$) instantiated rules. On DBpedia3.8, the proportion is also over 90%. Although we can argue the large contribution of overfitting rules made by long instantiated rules is attributed to the fact that ORP_{or} is proportional to RP_{all} and long instantiated rules often have large RP_{all}, long instantiated rules nevertheless make up the largest portion of overfitting rules. By comparing ORP_{type} with smooth confidence between CAR and instantiated rules on both datasets, we observe that on DBpedia3.8 without validation, the instantiated rules of all lengths have a 76% average probability of being overfitting, whereas CAR has 52%. On Wikidata, the probability of instantiated rules becomes 82% compared to 23% of CAR. Considering long instantiated rules are not only larger in size than CARs but also more likely to be overfitting, it is more important for instantiated rule learners to identify and remove overfitting rules than learners that only mine abstract rules.

To answer questions 3 and 4, we analyze Table.3.5 vertically. We consider a quality measure better than another if it has smaller overfitting proportion while maintaining a larger rule space. By this criterion, the smooth confidence outperforms standard confidence and PCA in that without validation, smooth confidence has average 91.56K rules and 83% ORP_{all} over both datasets, whereas standard confidence has 87.16K and 88%, and PCA has 84.28K and 93%. The advantage of smooth confidence becomes even more evident when validation is applied, where it has 11.85K rules and an overfitting proportion of 46% in comparison to 7.5K and 51% with standard confidence and 4.3K and 64% with PCA. One perspective to evaluate the effectiveness of the validation method is to compare the difference in overfitting proportion before and after validation. As shown in Figure.3.4, by changing the overfitting factor from 0 to 0.1, the overfitting proportion drops dramatically. With increasing factor, it recovers gradually partially because with higher factor, the filter removes more rules yet the overfitting rules that have high validation precision remain untouched, thus the overfitting proportion increases.

		DBpedia3.8													
		All		CAR			len=1			len=2			len=3		
Measure	Validation	Rules	ORP_{all}	RP_{all}	ORP_{or}	ORP_{type}	RP_{all}	ORP_{or}	ORP_{type}	RP_{all}	ORP_{or}	ORP_{type}	RP_{all}	ORP_{or}	ORP_{type}
Standard	Yes	6.03K	0.611	0.168	0.111	0.406	0.089	0.080	0.548	0.743	0.810	0.666	0	0	0
	No	96.29K	0.915	0.019	0.011	0.519	0.082	0.083	0.922	0.841	0.858	0.934	0.058	0.049	0.780
Smooth	Yes	11.06K	0.499	0.096	0.074	0.386	0.057	0.060	0.526	0.608	0.719	0.590	0.239	0.144	0.302
	No	98.91K	0.860	0.019	0.012	0.521	0.077	0.081	0.906	0.823	0.864	0.903	0.081	0.043	0.457
PCA	Yes	3.34K	0.736	0.071	0.053	0.548	0.067	0.054	0.597	0.862	0.893	0.762	0	0	0
	No	90.54K	0.952	0.014	0.013	0.837	0.091	0.091	0.953	0.870	0.877	0.960	0.024	0.017	0.653

		Wikidata													
		All		CAR			len=1			len=2			len=3		
Measure	Validation	Rules	ORP_{all}	RP_{all}	ORP_{or}	ORP_{type}	RP_{all}	ORP_{or}	ORP_{type}	RP_{all}	ORP_{or}	ORP_{type}	RP_{all}	ORP_{or}	ORP_{type}
Standard	Yes	9.14K	0.402	0.038	0.015	0.159	0.030	0.035	0.469	0.429	0.557	0.522	0.503	0.557	0.445
	No	78.03K	0.852	0.007	0.002	0.286	0.061	0.067	0.934	0.584	0.619	0.903	0.348	0.313	0.767
Smooth	Yes	12.65K	0.439	0.040	0.011	0.126	0.031	0.039	0.545	0.508	0.626	0.541	0.421	0.326	0.340
	No	84.22K	0.803	0.008	0.002	0.236	0.057	0.064	0.898	0.557	0.583	0.840	0.378	0.351	0.747
PCA	Yes	5.36K	0.563	0.024	0.009	0.209	0.037	0.036	0.541	0.564	0.575	0.574	0.375	0.380	0.571
	No	78.02K	0.917	0.006	0.004	0.713	0.061	0.063	0.949	0.583	0.590	0.928	0.350	0.342	0.897

Table 3.5 Experiment results for overfitting analysis. "All", "CAR", "$len = 1$", "$len = 2$" and "$len = 3$" are the domain of t in ORP_st. For instance, ORP_type under the "CAR" column measures the proportion of overfitting CARs to all CARs.

	FB15K-237				WN18RR			
Algorithm	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
DistMult [136]	0.241	0.419	0.263	0.155	0.430	0.490	0.440	0.390
ComplEx [123]	0.247	0.428	0.275	0.158	0.440	0.510	0.460	0.410
ConvE [26]	0.316	0.491	0.350	0.239	0.460	0.480	0.430	0.390
R-GCN+ [107]	0.249	0.417	0.264	0.151	-	-	-	-
TuckER [5]	0.358	0.544	0.394	0.266	0.470	0.526	0.482	0.443
RotatE [120]	0.338	0.533	0.375	0.241	0.476	0.571	0.492	0.428
QuatE [144]	0.366	0.556	0.401	0.271	0.488	0.582	0.508	0.438
AMIE+ [39]	-	0.409	-	0.174	-	0.388	-	0.358
Neural LP [137]	0.240	0.362	-	-	0.435	0.566	0.434	0.371
RuleN [76]	-	0.420	-	0.182	-	0.536	-	0.427
AnyBURL [75]	0.301	0.484	0.341	0.227	0.471	0.537	0.488	0.442
GPFL-ins0-car3	0.253	0.421	0.285	0.189	0.455	0.529	0.475	0.423
GPFL-ins1-car3	0.315	0.498	0.355	0.241	0.479	0.552	0.499	0.448
GPFL-ins2-car3	0.283	0.459	0.318	0.214	0.471	0.541	0.486	0.443
GPFL-ins3-car3	0.277	0.448	0.311	0.209	0.453	0.499	0.465	0.433
GPFL-Ensemble	0.322	0.504	0.362	0.247	0.480	0.552	0.500	0.449

Table 3.6 KGC experiment results in default setting. The top section contains embedding-based methods; the middle section includes logic-based methods, and the bottom section reports the GPFL results with various configurations. Best results in each section are underlined.

Correspondingly in Table.3.5, we also observe significant drops in ORP_{all} and ORP_{type} when the validation method is applied. At last, we investigate the effect of the removal of overfitting rules on predictive performance. As shown in Figure.3.3, the GAP of top-50 rules increases from near 0 to around 20% with overfitting factor being set from 0 to 0.1. In Table.3.4, we also observe significant improvements by having validation applied to both datasets. As without validation, the precision of top rules is extremely bad. We argue that the impact of overfitting rules, especially on instantiated rule learners, is too significant to ignore and must be handled properly.

In conclusion, we observe that long instantiated rules make up the largest portion of overfitting rules; instantiated rules are more likely to be overfitting than abstract rules; the choice of quality measure considerably affects the overfitting proportion, and the smooth confidence is better than standard confidence and PCA according to our criterion; the proposed validation method can effectively filter out a large portion of overfitting rules, and the predictive performance improves significantly with validation applied. The removal of overfitting rules improves the generalization ability of rule models. The generalization ability of a model is decided by its ability to extract the underlying patterns and to differentiate real patterns from noises. GPFL extracts better patterns by allowing the creation of instantiated rules, yet makes the differentiation of noises harder. With the proposed validation method, GPFL shows better generalization ability than existing systems as demonstrated in the following section.

	FB15K-237				WN18RR			
Algorithm	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
AnyBURL	0.269	0.452	0.311	0.193	0.288	0.345	0.302	0.263
AnyBURL-valid	0.273	0.458	0.323	0.199	0.291	0.352	0.305	<u>0.265</u>
GPFL-ins3-car3-valid	<u>0.283</u>	<u>0.469</u>	<u>0.328</u>	<u>0.205</u>	<u>0.294</u>	<u>0.361</u>	<u>0.311</u>	0.263
GPFL-ins3-car3	0.249	0.432	0.288	0.177	0.257	0.292	0.266	0.242

Table 3.7 KGC results in random setting. The "valid" suffix means the system is executed with validation applied. Best results are underlined.

3.3.6 Knowledge Graph Completion

A knowledge graph completion (KGC) query takes the form of $r_t(e_i,?)$ or $r_t(?,e_i)$ where r_t is the target predicate, and the question mark is expected to be replaced with candidates $e \in \mathscr{E}$ that are suggested by the learned rules such that predictions $r_t(e_i,e)$ or $r_t(e,e_i)$ for target r_t are proposed. We follow the evaluation protocol used in [13] to evaluate GPFL on KGC task. GPFL answers both head queries $r_t(?,e)$ and tail queries $r_t(e,?)$ that are created by corrupting the test triples. We use hits@1, hits@3, hits@10 and Mean Reciprocal Rank (MRR), all in the filtered setting, to report the experiment results. As a prediction can be suggested by multiple rules, it introduces complexity in ranking the predictions. In this work, we use the maximum aggregation strategy proposed in AnyBURL [75] to rank predictions. In particular, predictions are sorted by the maximum confidence of rules suggesting the predictions, and if there are ties among predictions, the tied predictions are sorted by recursively comparing the next highest confidence of suggesting rules until all ties are resolved.

We select FB15K-237 and WN18RR for KGC evaluation. As GPFL allows fine-tuning on the composition of learned rule types, we evaluate GPFL over various rule composition configurations. Specifically, the notation "ins*A*-car*B*" is used to indicate the maximal length of instantiated rules as *A* and that of CARs as *B*. For instance, "ins0-car3" depicts a configuration that only learns CARs of maximum length of 3, whereas GPFL with "ins3-car3" learns both instantiated rules and CARs of maximum length of 3. As we observed that the predictive performances for target predicates differ significantly with different configurations, we created an ensemble mode that aggregates the best performing rules of each target to form an optimal rule set. We test GPFL in two data settings: 1.) the default setting where we use the default data splits of FB15K-237 and WN18RR downloaded from AnyBURL's repository[2], and 2.) the random setting where we re-split the original dataset into training/validation/test sets in a 6:2:2 ratio. This is because to make the proposed validation method effective, we need much larger validation sets than the ones with the original FB15K-237 and WN18RR splits. As shown in Table.3.6, GPFL-ins1-car3 and GPFL-Ensemble demonstrate competitive performance compared to many of the state-of-the-art embedding algorithms and all of the logic-based methods, including AnyBURL. Noticeably, the performances of GPFL systems, such as GPFL-ins2-car3 and GPFL-ins3-car3, deteriorate with the inclusion of long instantiated rules. This is because of the inclusion of the overfitting rules that are mostly made up by long instantiated rules. This assumption is

[2]http://web.informatik.uni-mannheim.de/AnyBURL/

	FB15K-237				WN18RR			
Rule Composition	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
CAR Only	0.226	0.401	0.243	0.175	0.241	0.338	0.263	0.208
HAR Only	0.081	0.142	0.102	0.063	0.074	0.111	0.092	0.057
BAR Only	0.192	0.356	0.209	0.162	0.204	0.281	0.226	0.163
All	0.283	0.469	0.328	0.205	0.294	0.361	0.311	0.263

Table 3.8 KGC results under different rule composition settings. Experiments are tested on the random setting with GPFL-ins3-car3-valid.

confirmed by the experiment results in Table.3.7 where GPFL and AnyBURL are evaluated in the random setting. With the application of the proposed validation method, considerable increases in performance are observed with GPFL, whereas AnyBURL had minor improvements due to its inability to mine long rules. In Table.3.8, we observe that CARs and BARs contribute the most to the predication accuracy, whereas the improvement attributed to HARs is minor. Although logic-based methods are outperformed by some of the recent embedding-based models, most embedding models are only capable of performing transductive inference while logic-based ones can provide inductive and interpretable inference.

The interpretability of GPFL is achieved through a rule application procedure. For abstract rule learners, such as AMIE+ [63] and RuDIK [91], the rule application procedure is straightforward as the number of mined rules is often small. Specifically, given a set of queries and learned rules, the rule learning procedures utilized in previous works ground every learned rule over the background knowledge to produce a set of new facts that answer the queries. As a result, each new fact is associated with its triggering rules and these rules serve as evidences that justify the proposition of the new fact. However, for learners that mine instantiated rules, such as GPFL and AnyBURL [74], the large volume of mined rules makes it impossible to ground every rule. Therefore, two rule application steps are introduced in GPFL to allow the rule application procedure to be finished in a reasonable time. First, a filtering step that removes irrelevant rules by matching the head constants in queries and rules is installed. For instance, given a query $r_t(e_1, ?)$ and rules $r_t(e_1, Y) \leftarrow r_1(Y, e_2)$ and $r_t(e_2, Y) \leftarrow r_1(Y, e_1)$, the second rule is irrelevant to the given query. When the number of remaining rules is still too large, we set a time constraint on the rule application procedure. For experiment results reported in Table.3.6 and Table.3.7, the time constraint is set to 1000 seconds. For different datasets, the choice of time constraint is decided by the requirements of execution time and prediction accuracy.

3.4 Conclusion

In this chapter, we presented the GPFL system, a probabilistic rule learner optimized to mine instantiated rules. In comparison to abstract rule learners, the inclusion of instantiated rules significantly improves a system's predictive power and interpretability. To overcome the problems appeared in AnyBURL that prevent the system to extract complex rules in an efficient manner, GPFL utilizes a novel two-stage rule learning mechanism that enables the collective rule evaluation and optimized saturation satisfaction, which leads to significant reduction in rule evaluation time and easy mining of complex rules compared to AnyBURL. Although instantiated rules bring many benefits to a rule learning system, they are more prone to be overfitting than abstract rules in that the quality of instantiated rules are often decided by a small number of instances and can be greatly affected by noise. Therefore, it is important to identify and remove overfitting instantiated rules. Through experiments, we demonstrated that complex instantiated rules have greater tendency than abstract or simple ones to be overfitting, and negatively affect the prediction accuracy. We introduced a simple validation method to remove overfitting rules and observed improvements in prediction accuracy. One important future direction of research is the design of effective methods that differentiate valuable rules from irrelevant or noisy ones. Although it is already an active research area, most of the recent researches focus on incorporating KG embeddings into the rule quality measure instead of devising novel end-to-end methods that jointly learn rules and embeddings. Reinforcement learning is one promising framework to achieve such goals.

Chapter 4

Rule Hierarchy Framework

In the previous chapter, we have introduced the rule generation and evaluation strategies employed by GPFL to make the mining of instantiated logical rules more efficiently than existing works. However, the overhead caused by the generation and evaluation of unpromising rules is left undiscussed. Existing bottom-up learners employ only Flat Pruning Methods (FPMs) that set rule quality thresholds to filter out unpromising rules. Because of the unavailability of rule hierarchies, the more effective Hierarchical Pruning Methods (HPMs) used in top-down learners are not utilized by bottom-up systems. In this chapter, we introduce a generic Rule Hierarchy Framework (RHF) that leverages a collection of novel subsumption frameworks to build proper rule hierarchies from the rules produced by bottom-up learners. Then, the rule hierarchies can be used to enable the application of HPMs to bottom-up learners for the better pruning of unpromising rules. As a case study, we adapt RHF and two HPMs inspired by top-down learners to GPFL and conduct extensive experiments on four public benchmark datasets. We show that the application of HPMs effectively removes unpromising rules, leading to significant reductions in the runtime and the number of learned rules, without compromising predictive performance.

4.1 Introduction

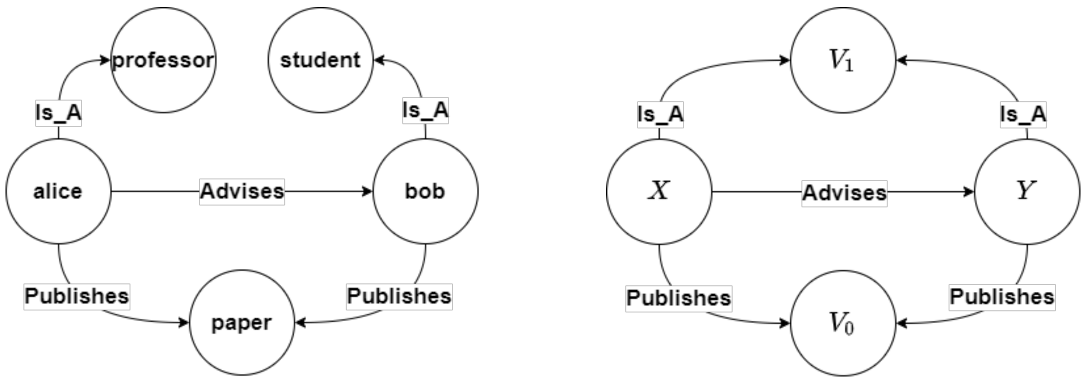
Rule learning systems often waste a large number of computational resources on creating and evaluating unpromising rules. For learners that only produce abstract rules, this issue is not apparent as the size of the rule space is often small. Thus the number of unpromising rules is neglectable. However, for instantiated logical rule learners that produce rules in the number of millions, the presence of unpromising rules severely affects the system's performance and scalability. Therefore, rule pruning techniques are in urgent need. Two types of rule pruning techniques are widely employed in existing works. Flat Pruning Methods (FPMs) filter out rules by checking against a set of pre-defined rule quality thresholds, and Hierarchical Pruning Methods (HPMs) make use of rule hierarchy, which contains the subsumption relationships between rules, to prevent the creation of descendant rules if their ancestors are considered unpromising by some criterion. HPMs are mostly employed by top-down learners that generate rules by simultaneously constructing and exploring a rule hierarchy. By contrast, because bottom-up learners generate rules by abstracting randomly sampled paths in Knowledge Graphs (KGs), the rule hierarchies of the generated rules are not readily available. As a result, the HPMs successfully employed in top-down learners can not be conveniently applied to bottom-up systems. In this chapter, we propose the Rule Hierarchy Framework (RHF) that efficiently builds a proper rule hierarchy from a set of learned rules by leveraging the properties of logical rules to simplify the process of deciding the subsumption relationships between rules. Unlike most existing works using exponentially complex subsumption frameworks to resolve subsumption relationships, we propose a position-constrained subsumption framework with linear complexity. Moreover, by further specifying the proposed subsumption framework, a proper rule hierarchy which contains no redundant subsumption relationships inferred via transitivity can be efficiently constructed from rules mined by bottom-up instantiated rule learners to scale the systems up through rule pruning.

Our contributions in this work are summarized as follows:

- We propose the RHF which is the first work that aims to build proper rule hierarchies from rules mined by bottom-up learners.
- We adapt RHF to GPFL where we design and implement two HPMs to effectively remove irrelevant and redundant rules.
- We demonstrate the effectiveness of the application of HPMs through experiments on four public benchmark datasets.

4.2 Related Work

In this section, we review some of the top-down and bottom-up systems, where we focus on the discussion about what and how rule pruning methods, including both flat and hierarchical ones, are implemented in existing works. FPMs that remove a rule if the rule fails some criterion are used extensively in both top-down and bottom-up methods to control the number and quality of mined rules. We here take the FPMs used in bottom-up learners for example. PRA [65] uses precision and coverage as criteria to filter out unqualified rules, and in RuleN [76] and AnyBURL [75], a rule is pruned if it is evaluated as unsatisfactory in terms of confidence and its coverage of positive examples. HPMs aim to prune a rule and its descendants in a hierarchy if the rule fails some criterion. The main benefit of HPMs over FPMs is that the pruning decision on a rule can be made by merely examining the ancestors of the rule, which makes the pruning of unpromising rules even before their creation possible. By avoiding the creation and evaluation of unpromising rules, a considerable amount of computational resources are saved. We take the HPMs used in top-down learners for example. QuickFOIL [140] prevents the creation of a rule and its specializations if the rule is considered as a syntactical duplicate; AMIE3 [63] makes a hierarchical pruning decision on a rule if the rule covers the insufficient amount of positive examples; ScaLeKB [21] uses a type of functional constraint to trigger the pruning, and RuLES [54] uses a hybrid quality measure that takes both statistical measures and a measure computed



(a) Knowledge graph (b) Abstraction

Fig. 4.1 A small knowledge graph and its abstraction.

over embeddings into consideration as the pruning criterion. Because of the unavailability of rule hierarchies in bottom-up learners, the more effective HPMs can not be conveniently adopted. Therefore, we need to first build a rule hierarchy from the rules mined by bottom-up learners, and then implement the HPMs to achieve improved efficiency.

4.3 Rule Hierarchy Framework

In this section, we present a Rule Hierarchy Framework (RHF) that constructs a proper rule hierarchy from a set of rules, where two main problems are addressed: 1.) how to efficiently decide the subsumption relationships between rules and 2.) what subsumption relationships to include in the rule hierarchy. We achieve this by proposing a collection of subsumption frameworks and an approach to compose a proper rule hierarchy that contains no redundancies.

4.3.1 Preliminaries

We here briefly review the notations used in the previous chapter and re-define some of them in the new context. A Knowledge Graph (KG) $\mathscr{G} = (\mathscr{E}, \mathscr{R}, \mathscr{T})$ is a directed multi-graph that represents relationships as triples denoted by $r(e_i, e_j) \in \mathscr{T}$ where $r \in \mathscr{R}$ is a relationship type and $e_i, e_j \in \mathscr{E}$ are entities. As we use clausal logic for knowledge representation and reasoning, we also call a relationship $r(e_i, e_j)$ a ground atom that is composed of a predicate r and constants e_i and e_j. The task of mining high-level patterns from KGs can be formulated as searching for first-order logic rules that express the regularities explaining the concepts presented in the KGs. In particular, the concepts are represented as predicates, and intuitively, the rules abstracted from the paths originated from the instances of the predicates are considered as the regularities. A path or a propositional rule, denoted by:

$$r_t(e_0, e_1), r_1(e_1, e_2), ..., r_n(e_n, e_{n+1})$$

is a sequence of ground atoms that starts from an instance of a target predicate r_t. In the formalism of definite Horn clause, we can rewrite the path as:

$$r_t(e_0, e_1) \leftarrow r_1(e_1, e_2), ..., r_n(e_n, e_{n+1})$$

which expresses a pattern where if body atoms $r_1(e_1, e_2), ..., r_n(e_n, e_{n+1})$ can be found in a KG, the existence of head atom $r_t(e_0, e_1)$ in the KG can be inferred.

Example 4.3.1. *Illustrated in Figure.4.1a is a small knowledge graph. Consider we want to find primitive patterns for the target predicate "Advises", and an instance of the predicate is $Advises(alice, bob)$ which states the fact that "alice advises bob", by traversing over its neighbourhood, we can extract a set of propositional rules as the primitive patterns. For instance, an extracted propositional rule is:*

$$p_1 : Advises(alice, bob) \leftarrow Publishes(alice, paper), Publishes(bob, paper)$$

which suggests that as alice and bob have published a paper together, the statement "alice advises bob" can be inferred.

4.3.2 Properties of Logical Rules

By replacing constants with variables in a propositional rule, we can convert it into first-order logic rules. We consider a rule *closed*, such as p_1, if all of the terms, which can be variables or constants, in the head atom also occur in the body atoms, otherwise a rule is considered *open*. A rule is *instantiated* if it contains at least one constant, otherwise it is an *abstract* rule. Now, we define three types of first-order logic rules that are considered useful in existing works, as follows:

$$\begin{aligned} \textbf{CAR}: &\quad r_t(X,Y) \leftarrow r_0(X,V_1), ..., r_n(V_n,Y) \\ \textbf{OAR}: &\quad r_t(X,Y) \leftarrow r_0(X,V_1), ..., r_n(V_n,V_{n+1}) \\ \textbf{IR}: &\quad r_t(X,Y) \leftarrow r_0(X,e_i), ..., r_n(V_n,e_j) \end{aligned}$$

where CAR and OAR stand for Closed Abstract Rule and Open Abstract Rule, respectively, and Instantiated Rules (IRs) are open rules that contain at least one constant. An OAR is also known as a template in Chapter.3. For an IR p, we use the deduction level, denoted by $d(p)$, to indicate the number of constants it contains. By convention, we use upper-case letters for variables and lower-case letters for constants, and symbols X and Y are reserved for the variables in the head atom.

Example 4.3.2. *From Figure.4.1a, in addition to closed rule p_1, we can also extract open rules:*

$$p_2 : Advises(alice, bob) \leftarrow Is\_A(alice, professor)$$
$$p_3 : Advises(alice, bob) \leftarrow Is\_A(bob, student)$$

and by replacing all of or a part of constants with variables in rules p_1, p_2 and p_3, we have:

$$p_4 : Advises(X,Y) \leftarrow Publishes(X,V_0), Publishes(Y,V_0)$$
$$p_5 : Advises(X,Y) \leftarrow Is\_A(X,V_1) \quad p_6 : Advises(X,Y) \leftarrow Is\_A(Y,student)$$

where p_4 is a CAR abstracted from p_1; p_5 is an OAR converted from p_2, and p_6 is an IR abstracted from p_3 and $d(p_6) = 1$.

Unlike logical rules mined using classic Inductive Logic Programming (ILP) [82] approaches with relaxed syntactic biases, the form of the rules extracted from KGs is restricted in accordance with the ontology of the KGs and the nature of paths. In particular, we define two important properties about the logical rules generated from paths.

Definition 4.1 (Connectedness)**.** *As the paths in KGs are connected, in the sense that adjacent ground atoms share a constant, the rules that are generalized from the paths are also connected, where adjacent atoms are connected via a connecting term.*

The connectedness of rules is desirable in that it ensures the body atoms of a rule are tied to each other and to the head atom via a chain of connections.

Definition 4.2 (Straightness)**.** *A rule is considered straight if for any term t in the rule, t occurs at most twice.*

As in other works [75, 45], the straightness of rules prevents the generation of cycles and syntactical equivalence that compromises system performance. In addition, we restrict our discussion to the KGs that only contain binary predicates. The rule space of a rule learner is a set containing all possible rules that can be produced by the learner. We denote by $\mathscr{F}$ a set of rules or a rule space. We model a rule hierarchy as a set of subsumption relationships $\Phi = \{\phi_0, ..., \phi_n\}$ where a subsumption relationship $\phi = (p, p')$ implies that p subsumes p'. We also use notation $(\mathscr{F}, \preceq)$ to conveniently describe a hierarchy which contains the subsumption relationships resolved by a subsumption framework $\preceq$ over the rules in $\mathscr{F}$. To efficiently build a rule hierarchy from a set of rules, the subsumption frameworks with efficient proof procedure and the approach that constructs rule hierarchies with minimum redundancy are needed.

4.3.3 Efficient Subsumption Framework

In this section, we present a subsumption framework that leverages the connectedness and straightness of rules to simplify the proof procedure of a variant of the θ-subsumption [72] for improved efficiency.

θ-subsumption, as a decidable approximation of logical entailment, is one of the most important subsumption frameworks employed in ILP works. A rule p θ-subsumes rule p', denoted by $p \preceq_\theta p'$, iff:

$$\exists\theta : p\theta \subseteq p' \tag{4.1}$$

where θ is a substitution that replaces variables by terms. For instance, we have $p_5 \preceq_\theta p_6$ with $\theta = \{V_1 \backslash student\}$. θ-subsumption is inconsistent with our assumption that all mined rules are straight in that it allows different variables to refer to the same entities. For consistency, we instead employ θ-subsumption under Object Identity (OI-subsumption) [34], denoted by $\preceq_{OI}$. A rule p OI-subsumes rule p' iff $p \preceq_\theta p'$ and all variables in p after substitution refer to different entities.

Example 4.3.3. *Figure.4.1b illustrates an abstraction of the KG in Figure.4.1a. In Figure.4.1b, only CARs and OARs are explicitly demonstrated as paths while IRs are implied by replacing variables with corresponding constants. Given the target predicate "Advises", we extract rules:*

$$p_7 : Advises(X,Y) \leftarrow \quad p_8 : Advises(X,Y) \leftarrow Publishes(X,V_0)$$

where p_7 is known as the top rule that subsumes all of the rules having target predicate "Advises". Consider we have a rule set $\mathscr{F} = \{p_4, p_7, p_8\}$, rule hierarchy $(\mathscr{F}, \preceq_{OI})$ is then $\{(p_8,p_4),(p_7,p_8),(p_7,p_4)\}$.

The proof procedure of deciding whether a rule p OI-subsumes rule p' can be summarized as follows: p' is first grounded by having all of its variables replaced with the constants not occurring in p and p'. We denote by $S(p')$ the grounded p'. Then an atom in p, denoted by $p[i]$ where $[i]$ is an element accessor that returns the i-th atom in a rule, is tested for elimination, that is if $p[i]$ subsumes an atom in $S(p')$ with a substitution that assigns variables to different entities, $p[i]$ is eliminated from p. This process is recursively performed until all atoms in p are eliminated and then we conclude $p \preceq_{OI} p'$. If the elimination test fails, it backtracks to test $p[i]$ against another atom in $S(p')$, and if all comparable atoms in $S(p')$ fail the test, we conclude that $p \npreceq_{OI} p'$.

Example 4.3.4. *Given rules:*

$$p_9 : r_t(X,Y) \leftarrow r_0(X,V_0)$$
$$S(p_{10}) : r_t(c_0,c_1) \leftarrow r_1(c_0,c_2), r_0(c_2,c_3), r_0(c_3,c_4)$$

we want to know whether $p_9 \preceq_{OI} p_{10}$. We first apply $\theta = \{X \backslash c_0, Y \backslash c_1\}$ to p_9 such that $p_9[0]$ is eliminated because $p_9[0]\theta = S(p_{10})[0]$. With $p_9[1]\theta = r_0(c_0,V_0)$, the proof procedure first tests $p_9[1]\theta$ against $S(p_{10})[2]$ because they share the same predicate r_0, which fails because c_0 can not be reduced to c_2. Then, the proof procedure backtracks to test $p_9[1]\theta$ against $S(p_{10})[3]$, which also fails. Therefore, we conclude $p_9 \npreceq_{IO} p_{10}$.

The backtracking process introduces unnecessary complexities to the resolution of OI-subsumption among straight and connected rules. We propose a constrained version of OI-subsumption that simplifies the proof procedure by replacing the backtracking process with a position constraint. Also, it is sound and complete w.r.t OI-subsumption.

Definition 4.3 (SA-Subsumption)**.** *Sequence-aware subsumption (SA-Subsumption) is defined as: given rules p and p', we consider p SA-subsumes p', denoted by $p \preceq_{SA} p'$, iff:*

$$\forall i \in [0, |p|] : p[i]\theta = p'[i] \tag{4.2}$$

where notation $|\cdot|$ indicates the number of atoms in a rule, and variables substituted in θ refer to different entities.

In comparison to the proof procedure of OI-subsumption that backtracks when the elimination test fails, SA-subsumption only needs to check if atoms at the same positions in rules p and p' are the same after substitution to decide the subsumption relationship between p and p'.

Proposition 4.1. *On connected and straight rules, SA-subsumption is sound and complete w.r.t OI-subsumption.*

Proof. Consider we have connected and straight rules p and p' where $|p| \leq |p'|$, we want to know whether $p \preceq_{OI} p'$. We eliminate $p[i-1] = r_0(V_0, V_1)$ by matching $S(p')[i-1] = r_0(c_0, c_1)$ with $\theta = \{V_0 \backslash c_0, V_1 \backslash c_1\}$ where the next atom $p[i] = r_0(V_1, V_2)$ is instantiated into $p[i]\theta = r_0(c_1, V_2)$. Due to the straightness of rules, c_1 can only occur at most once in the rest of $S(p')$ except for $S(p')[i-1]$, and according to the connectedness of rules, c_1 can only occur in $S(p')[i]$. To eliminate $p[i]$, we need $p[i]\theta' = S(p')[i]$ or simply $p[i]\theta' = p'[i]$ where θ' is extended from θ. Therefore, we prove when $p \preceq_{OI} p'$, $p \preceq_{SA} p'$. One special case is that, with rules p_9 and:

$$p_{11} : r_t(X,Y) \leftarrow r_0(Y,V_0), ..., r_0(X,V_n) \quad p_{12} : r_t(X,Y) \leftarrow r_0(X,V_n), ..., r_0(Y,V_0)$$

we can prove $p_9 \preceq_{OI} p_{11}$ yet $p_9 \not\preceq_{SA} p_{11}$. We preserve the completeness by reversing the order of the body atoms in p_{11} to create an equivalent rule p_{12} that have $p_9 \preceq_{OI} p_{12}$ and $p_9 \preceq_{SA} p_{12}$. It is obvious that SA-subsumption is sound w.r.t OI-subsumption as when $p \preceq_{SA} p'$, we have $\exists \theta : p\theta \subseteq p'$, thus $p \preceq_{OI} p'$. □

4.3.4 Proper Rule Hierarchy

According to Proposition 4.1 and Example 4.3.3, we can infer that $(\{p_4, p_7, p_8\}, \preceq_{SA})$ is also $\{(p_8, p_4), (p_7, p_8), (p_7, p_4)\}$. As SA-subsumption is transitive, knowing (p_7, p_8) and (p_8, p_4), relationship (p_7, p_4) can be inferred via transitivity. We call a subsumption relationship *redundant* if it can be inferred via transitivity, and a rule hierarchy *proper* if it contains no redundant relationships. We here introduce an approach that builds a proper rule hierarchy Φ from a rule set $\mathscr{F}$ w.r.t SA-subsumption, that is $\forall (p, p') \in \Phi : p \preceq_{SA} p'$.

To build a proper rule hierarchy, we take inspirations from the top-down methods [39] that simultaneously construct and explore rule hierarchies by repeatedly applying atomic specialization operators. An atomic specialization operator produces a rule p' from another rule p such that $p \preceq p'$ by modifying at most one element in p' at a time. By employing the subsumption frameworks that only identify the subsumption relationships corresponding to the relationships

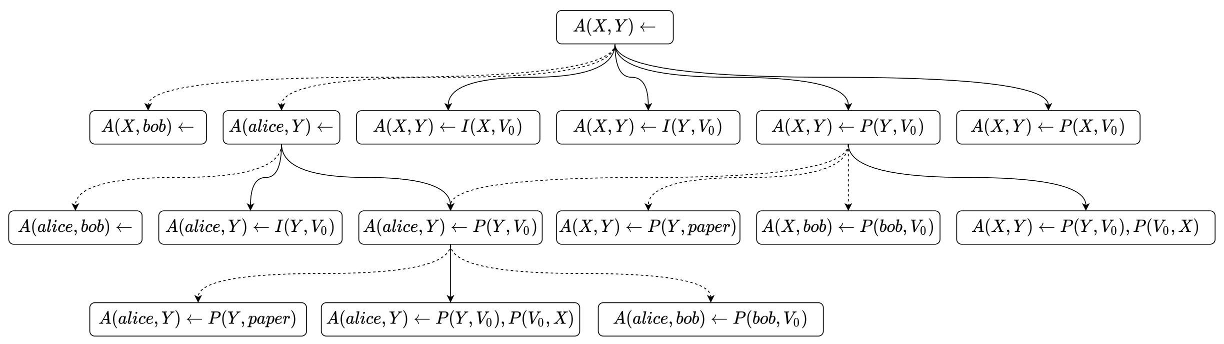

$A(X,Y) \leftarrow$
$A(X,bob) \leftarrow$
$A(alice,Y) \leftarrow$
$A(X,Y) \leftarrow I(X,V_0)$
$A(X,Y) \leftarrow I(Y,V_0)$
$A(X,Y) \leftarrow P(Y,V_0)$
$A(X,Y) \leftarrow P(X,V_0)$
$A(alice,bob) \leftarrow$
$A(alice,Y) \leftarrow I(Y,V_0)$
$A(alice,Y) \leftarrow P(Y,V_0)$
$A(X,Y) \leftarrow P(Y,paper)$
$A(X,bob) \leftarrow P(bob,V_0)$
$A(X,Y) \leftarrow P(Y,V_0),P(V_0,X)$
$A(alice,Y) \leftarrow P(Y,paper)$
$A(alice,Y) \leftarrow P(Y,V_0),P(V_0,X)$
$A(alice,bob) \leftarrow P(bob,V_0)$

Fig. 4.2 An incomplete proper rule hierarchy generated based on the knowledge graph in Figure.4.1a. We use abbreviations for lengthy predicate names, where *A* for "Advises", *I* for "Is_a" and *P* for "Publishes". Dashed lines represent I-subsumption relationships and solid lines represent A-subsumption relationships.

implied by atomic specialization operators, we can map a set of rules to a proper rule hierarchy. By assuming the continuity of rule sets, that is, for instance, if rule p_4 exists, its generalization p_8 also exists in the rule set, we here propose two subsumption frameworks that are derived from SA-subsumption and correspond to the atomic specialization operators that perform atom addition and variable instantiation, respectively.

Definition 4.4 (A-Subsumption)**.** *Addition subsumption (A-Subsumption) identifies the relationship between rules p and p' implied by a specialization operator which adds a new atom that shares a connecting variable with the last atom in p to p to create p'. p A-subsumes p', denoted by $p \preceq_A p'$, iff $p \preceq_{SA} p'$, $d(p) = d(p')$ and $|p'| = |p| + 1$.*

Definition 4.5 (I-Subsumption)**.** *Instantiation subsumption (I-Subsumption) identifies the relationship between rules p and p' implied by a specialization operator which instantiates a variable in p to create p'. p I-subsumes p', denoted by $p \preceq_I p'$, iff $p \preceq_{SA} p'$, $d(p) = d(p') + 1$ and $|p| = |p'|$.*

Given a rule set $\mathscr{F}$, as the individual hierarchies $\Phi_a = (\mathscr{F}, \preceq_A)$ and $\Phi_i = (\mathscr{F}, \preceq_I)$ are proper and resolved by the constrained versions of SA-subsumption, the combined hierarchy $\Phi = \Phi_a \cup \Phi_i$ is proper w.r.t SA-subsumption.

Example 4.3.5. *Illustrated in Figure.4.2 is the union of hierarchies ordered by A-Subsumption and I-Subsumption. It is a proper hierarchy w.r.t SA-subsumption. For instance, although the subsumption relationship $A(X,Y) \leftarrow \preceq_{SA} A(X,bob) \leftarrow P(bob,V_0)$ is valid, it is not explicitly linked in the hierarchy as it can be inferred via transitivity by considering $A(X,Y) \leftarrow \preceq_{SA} A(X,Y) \leftarrow P(Y,V_0)$ and $A(X,Y) \leftarrow P(Y,V_0) \preceq_{SA} A(X,bob) \leftarrow P(bob,V_0)$.*

4.4 Framework Adaptation

In this section, we adapt RHF to GPFL, and design and implement two HPMs, namely prior pruning and post pruning, that utilize generated rule hierarchies to remove irrelevant and redundant rules, respectively. GPFL is a discriminative rule learner that takes the bottom-up strategy

Algorithm 4.1: Augmented GPFL with Prior and Post Pruning

```
   Input  : $\mathscr{G}$, $I^+$, $len$
   Output : Mined rule set $\mathscr{F}$
1  Initialize empty set $\mathscr{F}$
2  $L \leftarrow$ Generalization($\mathscr{G}$, $I^+$, $len$)
3  $\Phi_a \leftarrow$ A-Subsumption($L$)
4  $L' \leftarrow$ PriorPruning($\Phi_a$)
5  for $l \in L'$ do
6      if $l$ is a CAR then
7          Check quality of $l$ and add $l$ to $\mathscr{F}$ if it is relevant
8      end
9      else
10         $S \leftarrow$ Specialization($l$, $\mathscr{G}$, $I^+$)
11         Filter out irrelevant rules in $S$
12         if $S$ is not empty after filtering then
13             $\Phi_i \leftarrow$ I-Subsumption($S$)
14             $S' \leftarrow$ PostPruning($\Phi_i$)
15             Add all rules in $S'$ to $\mathscr{F}$
16         end
17     end
18 end
19 Return $\mathscr{F}$
```

to generate abstract rules and then specializes OARs into IRs in a top-down manner. Two types of IRs are produced by GPFL, including:

$$\textbf{HAR}: \quad r_t(X,e_i) \leftarrow r_0(X,V_1),...,r_n(V_n,V_{n+1})$$

$$\textbf{BAR}: \quad r_t(X,e_i) \leftarrow r_0(X,V_1),...,r_n(V_n,e_j)$$

A Head Anchored Rule (HAR) is a specialization of an OAR where the non-connecting variable in the head atom is substituted with a constant, and a Both Anchored Rule (BAR) is a specialization of a HAR where the non-connecting variable in the last body atom is replaced by a constant. The two-stage rule generation mechanism allows GPFL to efficiently create and evaluate IRs where structurally similar IRs are collectively evaluated over shared groundings. It also presents an opportunity, by adopting RHF, to perform stage-wise hierarchical pruning to effectively remove unpromising rules.

In Algorithm.4.1, we introduce an augmented GPFL where we closely align the construction of rule hierarchies and the application of HPMs to the two-stage rule generation mechanism employed in GPFL. The system takes as inputs a knowledge graph $\mathscr{G}$, a set of positive instances of a target predicate I^+, and the maximum length a rule can have len and outputs a rule set $\mathscr{F}$. The length of a rule is the number of its body atoms. In the procedure `Generalization(`$\mathscr{G}$,I^+,len`)`, the system generalizes a set of paths originated from I^+ within length len into OARs and CARs and adds them to a rule set L. It then applies A-subsumption to L to build a rule hierarchy Φ_a that only contains the subsumption relationships between abstract rules.

Before proceeding to discuss the prior pruning, we review some of the rule quality indicators introduced in Chapter.3. Consider we have a rule p, the support [39] of p is defined as:

$$supp(p) = |H_p \cap I^+| \tag{4.3}$$

where H_p is the head grounding of p defined in Equation.3.1. We define the head coverage of a rule as:

$$hc(p) = \frac{supp(p)}{|I^+|} \tag{4.4}$$

and smooth confidence [75] as:

$$smc(p) = \frac{supp(p)}{\eta + |H_p|} \tag{4.5}$$

where η is an user-defined offset. We call a rule *p relevant* iff $supp(p) > supp_f$, $hc(p) > hc_f$ and $smc(p) > smc_f$ where $supp_f$, hc_f and sc_f are pre-defined thresholds for corresponding measures. We aim to use the procedure `PriorPruning(`Φ_a`)` to remove irrelevant IRs before their creation. More specifically, as IRs are generated by specializing OARs, the system can avoid the creation and evaluation of irrelevant IRs by identifying and pruning the OARs that potentially create irrelevant IRs. Therefore, the problem is reduced to the efficient identification of unpromising OARs. In the `PriorPruning(`Φ_a`)`, the system traverses the hierarchy Φ_a starting from the top rule, which is the rule that does not have any ancestors, in a breadth-first fashion. A visited rule p and all of its descendants are pruned if $supp(p) < supp_h$ where the prior threshold $supp_h$ is a pre-defined threshold on support. This makes sense because the support of rules is a monotonic measure that tends to become smaller with increasing depth in a rule hierarchy.

After the application of prior pruning, the system iterates over rules in the filtered rule set L'. For a rule l in L', if it is a CAR and considered relevant after evaluation, it will be added to the rule set $\mathscr{F}$. If l is an OAR, the procedure `Specialization(`l,$\mathscr{G}$,I`)` is applied where a set of HARs and BARs S are derived from l by instantiating certain variables in l. Irrelevant rules in S are then filtered out. We call an OAR *informative* if its S is not empty after the removal of irrelevant rules, otherwise it is *uninformative*. Inspired by the hierarchical feature selection approaches proposed in Ristoski and Paulheim [102], we utilize a simple mechanism to identify and prune redundant rules. Given a rule hierarchy $(\mathscr{F}, \preceq)$ over $\mathscr{G}$, we consider a rule *p redundant* if:

$$\exists p' \in \mathscr{F} : p' \preceq p, smc(p') > smc(p) \tag{4.6}$$

			#Triples			
Dataset	#Entities	#Predicates	#Train	#Valid	#Test	#Total
FB15K-237-LV	14.54K	237	185K	62.02K	62.12K	310K
NELL995-LV	75.49K	200	92.44K	30.84K	30.92K	154K
WN18RR-LV	40.94K	11	55.79K	18.60K	18.60K	93K
OBL-PE	180K	28	4.19M	183K	180K	4.55M

Table 4.1 Statistics of the benchmark datasets.

We argue that as $H_p \subseteq H_{p'}$ because of $p' \preceq p$, and $smc(p') > smc(p)$, the existence of p does not provide new information to the reasoning over $\mathscr{G}$. We corroborate the argument via the experiment results in the following section. By applying I-subsumption to S, the generated hierarchy Φ_i contains pairs of HARs and BARs that share subsumption relationships. In the procedure `PostPruning(`Φ_i`)`, by removing the BARs that have smaller confidence than the HARs subsuming them, the system creates a set S' free of redundant rules, and then adds all rules in S' to $\mathscr{F}$. Eventually, $\mathscr{F}$ is returned as the learned rule set where irrelevant and redundant rules are removed by the prior and post pruning, respectively.

4.5 Experiments

In this section, we use the augmented GPFL as an example to demonstrate the effectiveness of the application of HPMs through experiments on four publicly available datasets in two settings. By enabling the application of the prior and post pruning, we observe considerable reductions in the runtime and the number of learned rules without compromising the predictive performance.

4.5.1 Datasets

We select four publicly available datasets for experiments, including FB15K-237 [121], WN18RR [26], NELL995 [134] and OBL-PE [15]. FB15K-237, WN18RR and NELL995 are popular datasets for evaluating the performance of methods on Knowledge Graph Completion (KGC) task, and OBL-PE is a subset of OBL dataset [15] that only contains positive triples. FB15K-237, NELL995 and OBL-PE are created in such a way that their validation and test sets contain no triples that are the reverse of known triples in the training set. These reverse triples allow models with trivial rule $r_t(X,Y) \leftarrow r_t(Y,X)$ to perform exceptionally well, which makes it hard to understand the true performance of different approaches. WN18RR has around 35% of the triples in validation and test sets that are reverse triples.

As pointed out in the previous chapter, the performance of instantiated rule learners suffers greatly from the appearance of overfitting rules, and to remove overfitting rules through validation, large validation set is needed. However, the sizes of validation sets in the default splits of FB15K-237, NELL995 and WN18RR are too small to be useful for validation. Specifically, the validation triples to total triples ratio is 5% for FB15K-237, 0.3% for NELL995 and 3% for WN18RR. In

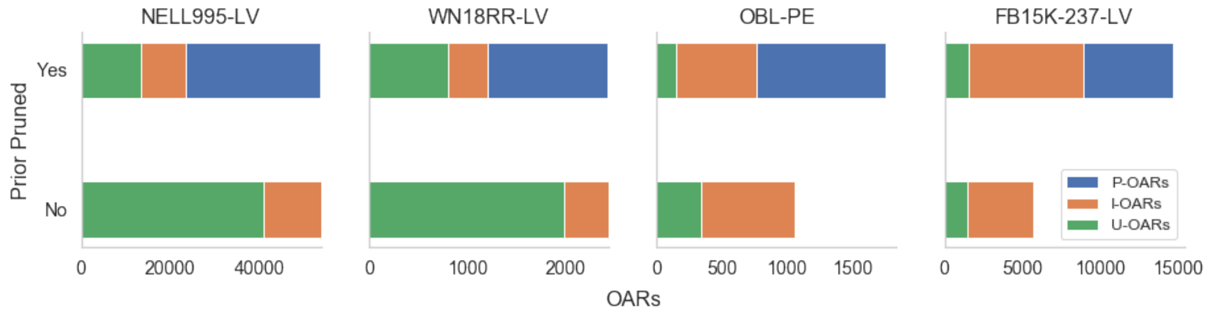


Fig. 4.3 Diagrams that show the numbers of different types of OARs in the sets of learned rules over experiment datasets, where P-OARs stands for pruned OARs; I-OARs for informative OARs, and U-OARs for uninformative OARs. The prior threshold is set to 10.

this work, we re-split FB15K-237, NELL995 and WN18RR into training/validation/test sets in a 6:2:2 ratio, and rename them by adding a "-LV" suffix that stands for large validation. Based on our observations, the performance of the re-split OBL-PE is similar to that of the original splits, so we keep the original OBL-PE for experiments. After re-splitting, the proportion of reverse triples is 6.5% for FB15K-237-LV, 19% for WN18RR-LV, and 6% for NELL995-LV. Statistics about these datasets is listed in Table.4.1.

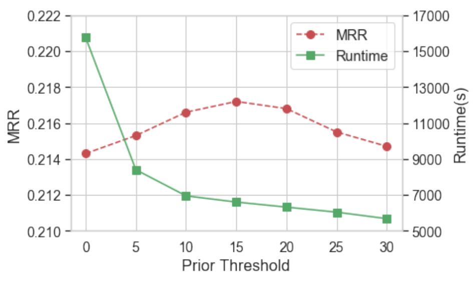
4.5.2 Experiment Setup

We implemented RHF and the HPMs on top of the GPFL codebase[1]. GPFL is implemented in Java and deeply integrated with the Neo4j[2] graph database. We configure GPFL to run in both constrained and unconstrained settings. For the majority of rule learners, it is often not possible to explore the entire rule space in a reasonable time on large KGs. Therefore, various time and space constraints are adopted to terminate systems prematurely to meet specific time and space requirements. For experiments on FB15K-237-LV and OBL-PF, we set time constraints on the generalization and specialization procedures, that is when the time constraints are reached, the system stops the running procedure and proceeds. For WN18RR-LV and NELL995-LV, we configure the system to run without constraints. For all experiments, we use the following setting of parameters: len for CARs and IRs is set to 3, and the filtering of overfitting rules is turned on where the overfitting threshold is set to 0.1. Except for WN18RR-LV, we set $supp_f$ to 3, hc_f to 0.001 and sc_f to 0.001. For WN18RR-LV, we set $supp_f$ to 2, hc_f to 0.0001 and sc_f to 0.0001. All experiments were conducted on the AWS EC2 instances that have 8 CPU cores and 64GM RAM. We have made our codebase and datasets available at https://github.com/irokin/RuleHierarchy.

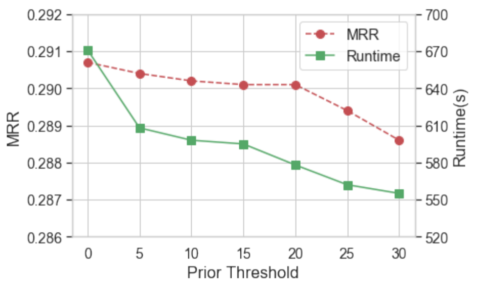
4.5.3 Evaluation Protocol

We evaluate the predictive performance of the system by tasking it with the KGC problem. Specifically, A KGC query takes the form of $r_t(e_i, ?)$ or $r_t(?, e_i)$ where r_t is the target predicate, and the question mark is expected to be replaced with the candidates $e \in \mathscr{E}$ that are suggested

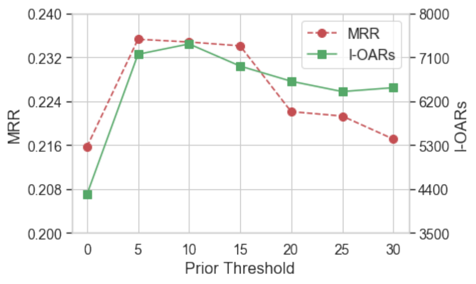
[1]https://github.com/irokin/GPFL
[2]https://github.com/neo4j/neo4j



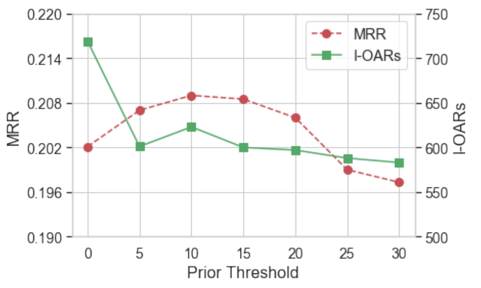
(a) NELL995-LV



(b) WN18RR-LV



(c) FB15K-237-LV



(d) OBL-PE

Fig. 4.4 Experiment results over different prior thresholds on the augmented GPFL. When the threshold is 0, it reports the results of the original GPFL.

by learned rules such that predictions $r_t(e_i,e)$ or $r_t(e,e_i)$ for r_t are proposed. We follow the evaluation protocol proposed in Bordes et al. [13] where both head query $r_t(e_i,?)$ and tail query $r_t(?,e_i)$ are answered. For ranking the predictions of a query, we adopt the maximum aggregation strategy proposed in Meilicke et al. [75] where predictions are sorted by the maximum of the confidence of rules suggesting the predictions, if there are ties, the tied predictions are resolved by recursively comparing the next highest confidence of suggesting rules until all ties are resolved. We report experiment results in Mean Reciprocal Rank (MRR) in filtered setting [13]. For FB15K-237-LV and NELL995-LV, we randomly select 20 target predicates for experiments, and for WN18RR-LV and OBL-PE, all predicates are used for experiments.

4.5.4 Prior Pruning

In this section, we show that the prior pruning is able to effectively remove irrelevant rules. All experiments are conducted without the application of the post pruning where when the prior threshold is set to 0 or turned off, the augmented GPFL is reduced to the original GPFL which serves as the baseline.

As prior pruning removes irrelevant rules by identifying and pruning unpromising OARs, and uninformative OARs (U-OARs) are the most unpromising OARs in that none of the rules derived from U-OARs are relevant, we use the number of U-OARs as an indicator to the effectiveness of the prior pruning. We start by discussing the experiment results on NELL995-LV and WN18RR-LV that are conducted in the unconstrained setting. As illustrated in Figure.4.3, on both NELL995-LV and WN18RR-LV, the numbers of U-OARs drop significantly while the numbers of informative OARs (I-OARs) stay more or less the same. The application of prior

Dataset	PostPrune	MRR	#R-Rules	RAT(s)
FB15K-237-LV	No	0.215	6.07M	4.17K
	Yes	0.216	5.11M	3.56K
OBL-PE	No	0.202	8.69M	2.75K
	Yes	0.202	7.63M	2.67K
NELL995-LV	No	0.214	5.59M	4.78K
	Yes	0.214	4.08M	4.27K
WN18RR-LV	No	0.291	28.9K	39.5
	Yes	0.291	28.8K	38.7

Table 4.2 Post pruning experiment results. #R-Rules is the number of relevant rules and RAT stands for the rule application time measured in seconds.

pruning succeeds in removing large portions of U-OARs on both datasets. Accordingly, as shown in Figure.4.4a and Figure.4.4b, the removal of U-OARs by the prior pruning results in significant decrease in the runtime where the predictive performance fluctuates within a small range. In the constrained setting, the effect of the removal of irrelevant rules is more complicated due to the interactions among various factors. In Figure.4.3, on both OBL-PE and FB15K-237-LV, the total number of visited OARs with the prior pruning applied is considerably larger than that of the baseline, which results in the discovery of more informative rules within a fixed time frame. This phenomenon is attributed to that the runtime, which would be wasted on creating and evaluating irrelevant rules if the prior pruning is turned off, is now allocated to further explore the rule space. In particular on FB15K-237-LV, the saved runtime leads to a 72% growth in the number of I-OARs, whereas on OBL-PE, most of the newly discovered rules are U-OARs. In Figure.4.4c and Figure.4.4d, as in constrained setting the runtime of the experiments with different thresholds is always the same, we instead compare the number of I-OARs. Specifically, we observe on FB15K-237-LV that the growth in I-OARs contribute to the improvements in the predictive performance, and on OBL-PE, because most of the I-OARs are already discovered with the baseline, its performance gain is negligible. It is worth noting that the prior pruning is much more effective on the datasets with large amount of predicates. This is reasonable because the prior pruning operates in the space of abstract rules and the amount of abstract rules is proportional to that of predicates.

4.5.5 Post Pruning

In contrast to the prior pruning that aims to remove irrelevant rules, the post pruning aims to remove relevant rules that are considered redundant. We demonstrate the validation of our definition of redundant rules and the effectiveness of the post pruning by examining if the predictive performance is affected by the removal of relevant rules. As demonstrated in Table.4.2, the removal of relevant rules has negligible impact on the predictive performance. One expected benefits from the removal of redundant rules is that the time used for rule application (RAT) decreases with the drops in the number of relevant rules. It is worth noting that on FB15K-237-

LV, 15% of the relevant rules are removed by the post pruning at the expense of a decrease by 0.01 in MRR, which consequently reduces the RAT by around 600s. Post pruning is much less effective on WN18RR-LV because most of its relevant rules have larger confidence than their ancestors.

4.5.6 Discussion

By indexing OARs and instantiated rules, the construction of rule hierarchies for prior and post pruning via A-subsumption and I-subsumption, respectively, is of linear complexity. The execution time and memory expense of prior and post pruning are neglectable compared to that of the learning of rules. It is ideal to turn on both prior and post pruning when the bottom-up learner is compatible. However, it is important to notice that, as prior pruning works on the rule hierarchy after the system has discovered all of the qualified OARs, it is reduced to a flat pruning method for learners that score OARs at the time of discovery. For post pruning, as it requires the rules in a hierarchy are already scored to work, it can be applied to any type of bottom-up rule learners.

4.6 Conclusion

In this chapter, we aim to apply HPMs to bottom-up instantiated rule learners for better predictive performance and scalability. To achieve this, we first introduced RHF that constructs a proper rule hierarchy from a set of rules using a collection of novel subsumption frameworks, and then adapted RHF to GPFL where we designed and implemented two HPMs that utilize the generated rule hierarchies to remove unpromising rules. Through experiments on four datasets with the augmented GPFL, we demonstrated the effectiveness of the application of HPMs, where we observed significant reductions in the runtime and the number of learned rules without compromising the predictive performance. This successful adaptation demonstrates the potential benefits and practicability when applying HPMs to bottom-up learners. By taking this work as a foundation, developing advanced HPMs that make use of the rule hierarchies produced by RHF are beneficial to the scaling up of existing bottom-up learners.

Chapter 5

Learning Logical Rules for Drug Repurposing

So far, we have introduced the development of systems and optimization approaches that enable the efficient mining of instantiated rules from large-scale Knowledge Graphs (KGs). In this chapter, by incorporating the techniques from previous chapters and state-of-the-art symbolic graph reasoning methods, we develop a novel drug repurposing system that takes a feature engineering strategy where a feature matrix with both abstract and instantiated rules as features and drug-disease pairs as rows is generated from biomedical KGs and then passed to train a conventional machine learning model for the prediction of potential drug-disease therapeutic associations. Through extensive experiments on two large-scale biomedical KGs, we demonstrate that the proposed system outperforms existing methods by a large margin. We also conducted a case study where we repurpose drugs for Pulmonary Arterial Hypertension (PAH) to show that the proposed system is capable of suggesting reasonable repurposing opportunities supported with interpretable evidences.

5.1 Introduction

According to a 2016 study [27], the process of bringing a new therapeutic drug to market is estimated to cost \$1.6 billion. Over the average development cycle of 15 years [100], the majority of candidate compounds selected by screening for biological targets fails to progress to clinical trials due to toxicity issue or low efficacy [50]. With the increase in development cost and time due to high attrition rate and the drop in the number of approved drugs in recent years [105], innovative approaches for efficacy prediction and target identification are in urgent need. Drug repurposing, a strategy for identifying new indications for approved or investigational drugs, offers fast and cost-effective solutions for therapeutic development [96]. The advantages of drug repurposing mainly attribute to the knowledge of existing drugs where the safety of these drugs has already been tested in clinical trials for other applications and their pharmacological analyses are often available for research [149]. The accelerated drug development cycle enabled by the repurposing strategy has made a significant impact on the

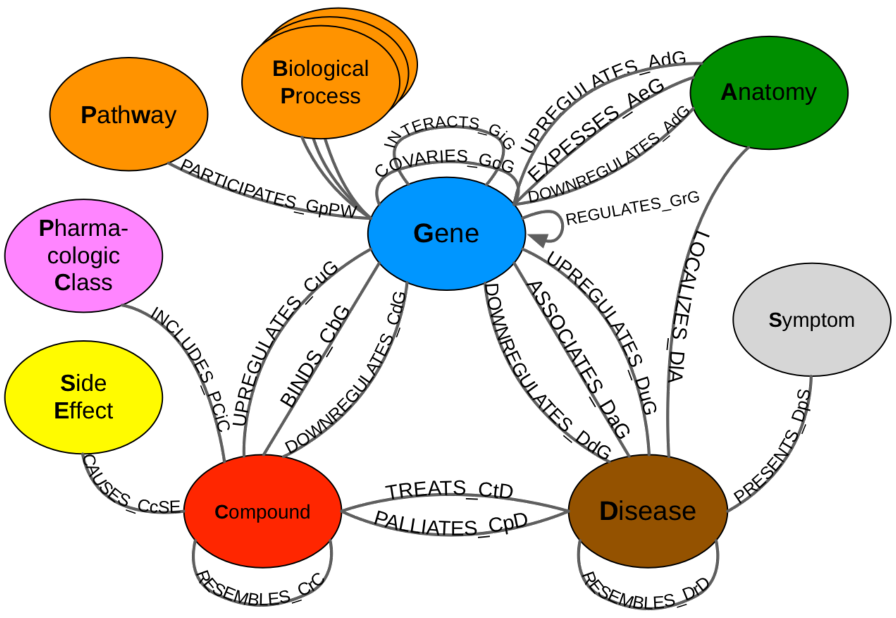


Fig. 5.1 The metagraph of the heterogeneous biomedical network Hetionet[1].

advancement of healthcare. Specifically, drug repurposing provides a feasible solution for finding therapeutic treatments for rare or emerging diseases where conventional drug discovery paradigms struggle to succeed [92]. There are over 8000 rare diseases affecting approximately 350 million people. Only 5% of rare diseases have authorized treatments. However, the low morbidity makes the investment in developing treatments for rare diseases through de novo drug discovery risky for large pharmaceutical companies [35]. Although with known concerns [14], drug repurposing allows for the therapeutic development of rare diseases to progress in a much faster pace with significantly less cost. For example, to test efficacy for Cushing's syndrome, the repurposed compound, mifepristone, required a cohort of fewer than 30 patients, whereas a new compound, levoketoconazole, required more than 90 participants [51]. Drug repurposing also plays an increasingly important role in the development of treatment for emerging diseases. A large body of research into repurposing existing drugs for COVID-19 [18] has been carried out in a short time [113, 62, 111]. Top-ranked candidates, including mefuparib [42], toremifene [147] and melatonin [148], are suggested by various systems based on machine learning algorithms and to be tested in clinical trials. With the acknowledgement of the significance of drug repurposing, research on developing novel drug repurposing systems has attracted increasing attention [96].

5.1.1 Data-driven Approaches

The rapid advancement in computing hardware and machine learning algorithms and the availability of a large amount of public biomedical databases have facilitated the development of data-driven approaches for the modelling of pharmacological processes [38]. In comparison to the experimental drug repurposing approaches relying on the screening of repositories of existing chemical compounds for the interactions with targets of certain diseases [125], Computational

[1]©Hetionet in Neo4j licensed under CC BY 4.0.

Drug Repurposing (CDR) aims to model drug efficacy for the prediction of new drug-disease treatment associations in a data-driven manner [55]. To such end, two fundamental assumptions underlying the design of CDR systems are established: 1.) because it is common for drugs to have multiple target proteins, it is reasonable that a multi-target drug might be used in multiple therapeutics, and 2.) a drug that acts on the shared phenotypic, genomic and clinical factors of different diseases might also be beneficial to more than one disease [153]. In the ground of the first assumption, many computational frameworks that extract insights from a single type of data have been proposed for drug repurposing. For instance, Zhao and Li [145] used protein target interaction networks for efficacy modeling, and Stanfield et al. [117] predicted the treatment relationship between diseases and drugs by consulting gene expression activation following different drug treatment regimes. However, given the complexity of the pathogenesis of diseases and the mechanism of action of drugs, no single data type can capture all of the factors necessary to understand the interplay that contributes to an effective therapeutics. Data integration has thus been proposed to account for the incompleteness of mining from a single type of source and support the realization of the second assumption by fusing multiple types of data for modelling [153]. The majority of CDR systems that adopt this idea models the integration of different types of data as a Knowledge Graph (KG) where biomedical entities are nodes and their interactions are links between nodes [52, 15, 57]. Modelling data from multiple sources as KGs opens up opportunities for the application of advanced graph reasoning systems [20] to drug repurposing task.

5.1.2 Reasoning on Biomedical Knowledge Graphs

Mining actionable insights from KGs for downstream analytical and predictive applications is the core task of knowledge graph reasoning. From the perspective of knowledge representation, we roughly classify relevant works into three groups: sub-symbolic methods [129] that aim to learn and reason with knowledge in the form of low-dimensional embeddings; symbolic systems [87] that represent knowledge as discrete symbols and leverage the principles of statistics and logic for learning, and hybrid approaches, such as neural-symbolic machines [53], that use sub-symbolic learning strategies to generate optimal symbolic outputs. Due to the easy access of implementations of various high-performance sub-symbolic graph reasoning frameworks, such as DGL [128], GraphVite [150] and BigGraph [68], a large number of recent CDR works utilize sub-symbolic systems to model drug efficacy. For instance, Sosa et al. [115] built a KG containing drug, disease and gene entities that are linked by weighted edges where the weight associated to an edge indicates the confidence score computed by mining biomedical literature, and an uncertain KG embedding method [19] is then applied for learning. Gysi et al. [48] identified 81 top-ranked repurposing candidates for COVID-19 by adopting the architecture of the graph neural network previously used for modelling polypharmacy side effects [152]. Although sub-symbolic works often demonstrate strong predictive performance, their black-box nature presents an additional challenge for risk-sensitive biomedical applications. Inability to meaningfully interpret mined patterns and understand correlations between graph regularities

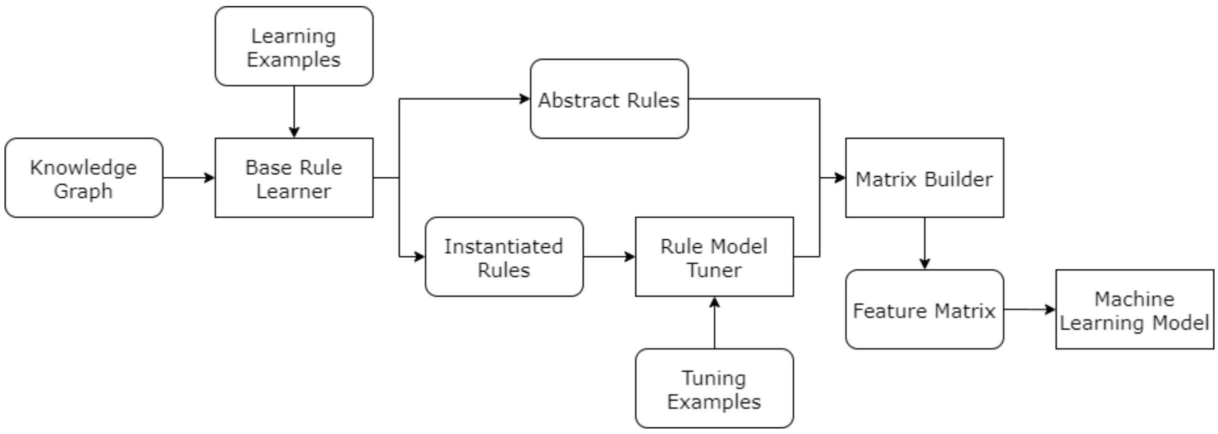


Fig. 5.2 Overview of Ranta system.

and task targets motivate researchers to investigate symbolic and hybrid methods that are more explainable. Noticeably, as most of the hybrid approaches, such as RLvLR [90], RUGE [47] and RuLES [54], are developed in the past 5 years, their applicability has not yet been explored for drug repurposing. In contrast, several symbolic systems have been proven effective in identifying and interpreting drug repurposing opportunities.

One representative symbolic system is the Rephetio project [52] where the authors first constructed a KG that encodes different types of entities and relationships extracted from millions of biomedical studies, and then built a feature matrix containing patterns that distinguish treatments from non-treatments. Specifically, a set of abstract rules (ARs) that describe the treatment relation between drug-disease pairs is derived from a manually declared metagraph. For instance, illustrated in Figure.5.1 is the metagraph of the KG built in Rephetio, where relationship type $TREATS\_CtD$ represents the concept of treatment and a possible AR that explains it is:

$$p_1 : TREATS\_CtD(X,Y) \leftarrow BINDS\_CbG(X,V_1), ASSOCIATES\_DaG(Y,V_1)$$

which states the pattern "if a compound X is bound to a gene V_1 and V_1 is associated to a disease Y, then X treats Y" where X, Y and $V1$ are variables that can be replaced by the concrete entities in the KG. The generated ARs are treated as columns of the feature matrix, and drug-disease pairs are the rows. For each cell, a specificity score indicating how likely a drug-disease pair is connected via an instantiation of an AR is computed by traversing all ground paths of the AR between the pair. Eventually, the filled feature matrix is utilized to train a logistic regression model that makes repurposing suggestions. Rephetio shows promising predictive performance under various evaluation settings, and most importantly, patterns strongly correlated to drug efficacy can be easily identified and interpreted through the ranking of AR features by model coefficients. However, the requirement for manually declared metagraph, the expensive computation of cell values via complete search, the inefficient implementation and the adoption of the top-down learning paradigm limit its practicability on learning from large-scale KGs.

5.1.3 Contribution

As experimentally demonstrated in AnyBURL [74] and GPFL, the inclusion of instantiated logical rules (IRs) for logical inference significantly improves predictive performance and system expressivity. Given that Rephetio also shows promising performance by implementing a feature engineering strategy with a degree-sensitive specificity score as feature values, it is reasonable to assume that a feature matrix complemented with IRs as additional features can further improve system performance. However, due to the enormous volume of IRs extracted from large-scale KGs, it is neither practical nor beneficial to include all IRs as features. An instantiated rule selection mechanism that only includes a small set of appropriate IRs as features is needed to address this challenge. In this work, we propose Ranta (Random Walk Based Metapath Mining), a symbolic knowledge graph reasoning system that integrates the ideas and implementations of multiple state-of-the-art systems. The overall design of Ranta is illustrated in Figure.5.2. A base rule learner is employed to generate both high quality ARs and IRs efficiently. The generated IRs are then passed to a rule model tuner where a minimal subset of IRs is selected by optimizing a loss function adopted from RuDIK [91] according to a set of tuning examples that decide what specific concepts to be modeled. Eventually, a matrix builder fills the matrix cells in with either Rephetio-style specificity scores or binary values. Similar to Rephetio and PRA [65], the generated matrix is then used to train a logistic regression model. Through experiments on two large-scale biomedical KGs for drug repurposing task with Ranta and recent symbolic systems, we observe that 1.) systems based on feature engineering strategy, including Ranta and Rephetio, outperform ones that rely on logic for inference, and 2.) with the inclusion of IRs, Ranta largely improves predictive performance over Rephetio where models that include multi-hop IRs (length$=2$) show better performance than ones with one-hop IRs (length$=1$). In summary, our contributions are:

- We propose a novel symbolic system that builds and learns from a feature matrix with both ARs and a subset of IRs as features.
- We conducted an empirical study where state-of-the-art symbolic systems are evaluated for drug repurposing task in terms of predictive performance and interpretability. We show that the proposed system outperforms existing alternatives.
- Through a case study that repurposes existing drugs for PAH, we show that Ranta is able to prioritize valid repurposing opportunities while providing reasonable explanations that support the predictions.

5.2 Methodology

In this section, we first introduce the necessary preliminaries and terminologies used to describe the system, and then elaborate on the design of Ranta. The introduction of Ranta is divided into three parts: the base rule learner in charge of producing both ARs and IRs efficiently; the rule model tuner that selects a subset of IRs that best represents the learning target, and the matrix builder which constructs a feature matrix from a given set of examples and the mined and selected logical rules.

5.2.1 Preliminaries

To better describe the system proposed in this chapter, we review some of the notations used in previous chapters and introduce new ones. A Knowledge Graph (KG) is denoted by $\mathscr{G} = (\mathscr{E}, \mathscr{R}, \mathscr{T})$ where a ground atom $r(e_i, e_j) \in \mathscr{T}$ is composed of a predicate $r \in \mathscr{R}$ and constants $e_i, e_j \in \mathscr{E}$. e_i is known as the subject of $r(e_i, e_j)$, and e_j the object. We denote by Sub_r and Obj_r the sets that contain all the subjects and objects of the ground atoms with predicate r in a KG, respectively. Given a non-ground and closed logical rule:

$$p_1 : r_t(X,Y) \leftarrow r_1(X,V_0), ..., r_n(V_n,Y)$$

we call $r_t(X,Y)$ the head atom and the rest the body atoms of p_1, and r_t is the target predicate and X and Y are the target variables. An instance of p_1 is:

$$p_2 : r_t(e_0,e_1) \leftarrow r_1(e_0,e_2), ..., r_n(e_n,e_1)$$

which is a path originated from $r_t(e_0,e_1)$. The instantiation relationship between p_1 and p_2 is denoted by $p_2 \in In(p_1)$. The instantiation of the target variables is called a head grounding, e.g., (e_0,e_1) is the head grounding of p_1 and denoted by h_{p_1}, and the corresponding body grounding is denoted by b_{p_1}. For a non-ground rule p, we define a set containing all the head groundings of p over a KG as:

$$H_p = \{(x,y) | (x,y) = h_{p'}, p' \in In(p)\} \tag{5.1}$$

Given a set of entity pairs E, we define the bounded coverage of a rule p as:

$$C_p(E) = \{(x,y) | (x,y) \in (H_p \cap E)\} \tag{5.2}$$

which contains the pairs in E covered by p. Following RuDIK [91] where the incomplete and noisy nature of KGs are taken into consideration, we further define the unbounded coverage of p as:

$$U_p(E) = \{(x,y) | (x,y) \in (H_{p^*} \cap E)\} \tag{5.3}$$

where p^* is a modified p such that the instantiation of target variables are only subject to their hosting body atoms instead of the connectivity of body atoms. For instance, p_1 is reduced to p_1^*:

$$p_1^* : r_t(X,Y) \leftarrow r_1(X,V_0), r_n(V_n,Y)$$

where $V_0 \neq V_n$ such that the instantiation of X and Y are decided only by their hosting body atoms. For a given open rule:

$$p_3 : r_t(X,Y) \leftarrow r_1(X,V_0),...,r_n(V_n,V_{n+1})$$

p_3^* then is:

$$p_3^* : r_t(X,Y) \leftarrow r_1(X,V_0)$$

where body atoms that contain no target variables are discarded. Given a rule p and a set E, we can observe that $C_p(E) \subseteq U_p(E)$. As shown in Figure.5.2, Ranta requires two types of examples to generate features: the learning examples passed to the base rule learner for the computation of rule quality and the tuning examples utilized by the rule model tuner to select an instantiated rule subset. For simplicity, we denote examples of the target predicate as I and use a super- and sub-scripts based convention to name different types of examples. Positive and negative examples are denoted by I^+ and I^-, respectively. Learning examples are denoted as I_l and tuning examples as I_u. For instance, I_l^+ represents positive learning examples and I_u^- means negative tuning examples. The definition and generation of these examples will be introduced in the following sections.

5.2.2 Ranta System

We formulate drug repurposing as a probabilistic binary classification problem where we aim to build a machine learning model that predicts the probability of whether there exists therapeutic relationship between a given drug-disease pair. In addition, it can also be described under the Knowledge Graph Completion (KGC) framework, that is to model the probability distribution of the existence of the treatment relationships between given entity pairs. Ranta approach this issue with a feature engineering strategy. Overall, Ranta assigns to each drug-disease pair a feature vector with both high quality ARs and a subset of IRs as features and specificity or binary scores as values. The logical rules are first produced by a base rule learner; then a small set of IRs, namely a rule model denoted by P, that best describes the target predicate in certain contexts is generated via a rule model tuner by optimizing a loss function; Both the ARs and rule model are then passed to a matrix builder that computes feature values, and eventually, the generated feature matrix is used to train a logistic regression model. In the rest of this section, we introduce in detail how this process is implemented in Ranta.

Base Rule Learner

A base rule learner takes as inputs a KG $\mathscr{G}$ and a set of positive learning examples I_l^+ of the target predicate r_t and produces logical rules that have head atoms with the target predicate. As discussed in Chapter.2, Rule learners differ in the choices of language bias, problem definition and learning paradigm. In the case of Rephetio, it generates a comprehensive set of Closed

Abstract Rules (CARs), e.g., p_1, in a top-down manner via the traversal of a manually declared metagraph. As a metagraph encodes both ontological and semantic information about the target concept to be modeled, the declaration of it requires domain knowledge. Moreover, the top-down learning mechanism tends to generate groundless rules causing system overheads. To address these issues, we instead use GPFL as the base learner in that, in addition to CARs, GPFL mines two types of IRs (Head Anchored Rules (HARs) and Both Anchored Rules (BARs)) in an efficient way from large-scale KGs. Detailed introduction and examination about GPFL are elaborated in Chapter.3.

Rephetio demonstrates that the inclusion of long CARs (maximal length of 4) as features yield the best outcome. To optimize the production of long CARs, we replace the uni-directional traverser implemented in GPFL with a bi-directional one [118] that starts the search from both ends of an instance. Noticeably, as GPFL uses random walkers to retrieve paths in the neighbourhood of instances for the generation of ARs, the setting of the number of random walkers has great impact on system performance. We review two quality measures used in Ranta for evaluation. The standard confidence of a rule p is defined as:

$$sc(p) = \frac{supp(p)}{H_p} \tag{5.4}$$

where $supp(p) = |C_p(I_l^+)|$ is the support of p, and the smooth confidence as:

$$smc(p) = \frac{supp(p)}{\eta + H_p} \tag{5.5}$$

with η being a pre-defined offset that mitigates the problem where rules with small H_p have better quality score yet often not necessarily more valuable than rules with large H_p. Noticeably, the computations of both $sc(p)$ and $smc(p)$ require only positive learning examples I_l^+ to compute the support, which implies the Closed World Assumption (CWA) is employed.

Rule Model Tuner

A rule model tuner takes in a list of IRs ranked by quality, denoted by P', and positive and negative tuning examples, and outputs a rule model P with $P \subseteq P'$. Positive tuning examples I_u^+ are provided by user and can be the same set as I_l^+ or a set customized for specific concepts. As I_u^+ is used to decide what rules to include in P, the choice of elements in I_u^+ has a significant impact on the resultant P, thus affects the system performance. It is rare for KGs to record true negative facts, and even in the case they do, the number of positive and negative facts is often largely imbalanced. Therefore, negative tuning examples I_u^- are, in most cases, generated by a negative sampling strategy. Here we introduce two negative sampling strategies evaluated in our experiments. The first one is the Local-closed World Assumption (LCWA) [39] which states

Algorithm 5.1: Rule Model Tuner

```
    Input  : $I_u^+, I_u^-, P', th, ti$
    Output : A rule model $P$
 1  Initialize an empty set $P$;
 2  for $p \in P'$ do
 3      if $\Delta l(p, P) < 0$ then
 4          $P \leftarrow p \cup P$;
 5          for $q \in (P \backslash p)$ do
 6              if $\Delta l(q, P \backslash q) \geq 0$ then
 7                  $P \leftarrow P \backslash q$;
 8              end
 9          end
10      end
11      if $l(P) \leq th$ or Timer() $> ti$ then
12          Break;
13      end
14  end
15  return $P$;
```

that, for a given target predicate r_t, the elements in the set:

$$I_{lcwa}^- = \bigcup_{x \in Sub_{r_t}} \{(x,y) | y \in \mathscr{E}, r_t(x,y) \notin \mathscr{T}\} \tag{5.6}$$

are considered negative examples. LCWA assumes the target predicate is functional, that is for a given subject x, the objects y in triples $r_t(x,y) \in \mathscr{T}$ are all possible entities that can be the objects for x in r_t. The RuDIK strategy [91] extends LCWA by assuming the functionality of the inverse of r_t and adopting a strict semantic constraint. We first define an auxiliary set:

$$I_{rudik_{obj}}^- = \bigcup_{y \in Obj_{r_t}} \{(x,y) | x \in \mathscr{E}, r_t(x,y) \notin \mathscr{T}\} \tag{5.7}$$

and then the set of negative examples produced by following the RuDIK strategy is:

$$I_{rudik}^- = \{(x,y) | (x,y) \in I_{lcwa}^- \cup I_{rudik_{obj}}^-, r'(x,y) \in \mathscr{T}, r' \neq r_t\} \tag{5.8}$$

where the condition $r'(x,y) \in \mathscr{T}$ with $r' \neq r_t$ states that a pair (x,y) must be linked by a predicate other than r_t to be considered negative, otherwise the entities in the pair are less likely semantically related. We drop the additional constraint requiring the subjects (objects) of the generated negative examples to be of the same entity type in the original RuDIK strategy. From our observation, it is common that entities of the types that are different but on the same branch of a type hierarchy occur at the same position in the instances of a predicate. For instance in Repotrial, entities of types "SmallMolecule" and "BiotechDrug" are sub-types of drug and appear as subjects in triples with "HasIndication" predicate.

Now we introduce a loss function adapted from RuDIK [91] that measures the fitness of a rule p to the target predicate r_t. The loss function is defined as:

$$l(p) = \alpha \cdot (1 - \frac{|C_p(I_u^+)|}{|I_u^+|}) + \beta \cdot \frac{|C_p(I_u^-)|}{|U_p(I_u^-)|} \tag{5.9}$$

where $\alpha, \beta \in [0,1]$ and $\alpha + \beta = 1$. $l(p)$ is composed of two components where the first component associated with α measures the proportion of positive examples not covered by the rule, and the second component indicates the degree of the presence of errors. For a rule model P, its loss function is then translated as:

$$l(P) = \alpha \cdot (1 - \frac{|C_P(I_u^+)|}{|I_u^+|}) + \beta \cdot \frac{|C_P(I_u^-)|}{|U_P(I_u^-)|} \tag{5.10}$$

Therefore, by tuning the composition of P such that $l(P)$ is minimized, a subset of IRs that best models the target predicate can be found. This can be achieved by iteratively adding rules $p \in P'$ that result in a decrease in $l(P)$ into P until a loss threshold th or a maximal runtime ti is reached. We define the change in $l(P)$ after the addition of a rule p as:

$$\Delta l(p,P) = -\alpha \cdot (\frac{|C_p(I_u^+) \backslash C_P(I_u^+)|}{|I_u^+|}) + \beta \cdot (\frac{|C_{p\cup P}(I_u^-)|}{|U_{p\cup P}(I_u^-)|} - \frac{|C_P(I_u^-)|}{|U_P(I_u^-|)}) \tag{5.11}$$

Given a rule p, we consider the addition of it to P beneficial when $\Delta l(p,P) < 0$. As shown in Algorithm.5.1, a rule $p \in P'$ is added into P if $\Delta l(p,P) < 0$. The for loop from line 5 to 9 acts as a replacement mechanism that is new to the original RuDIK work. The replacement mechanism is used to discard existing rules in P that become redundant or inappropriate because of the addition of a new rule. For instance, given a newly added rule p and an existing rule p', if $C'_p(I_u^+) \subset C_p(I_u^+)$ and $C_p(I_u^-) \subset C'_p(I_u^-)$, then the removal of p' after the addition of p is very likely to further optimize the loss of P.

Matrix Builder

Once a rule model P is produced, a matrix builder takes the union of P and ARs, denoted by $R = P \cup ARs$, and a set of known drug-disease pairs I to create a feature matrix $M \in \mathbb{R}^{|I| \times |R|}$ where a cell value $m_{ij} \in M$ measures the relevance of a logical rule $p_j \in R$ to a pair $(x,y)_i \in I$ and is computed by a feature value function $s : \mathscr{G}, p_j, (x,y)_i \rightarrow m_{ij}$. We select two feature value functions for evaluation. The first one is a simple yet proven function:

$$s(\mathscr{G}, p_j, (x,y)_i)_{bin} = \begin{cases} 1 & \exists p \in In(p_j) : h_p = (x,y)_i \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

which uses binary values to measure the existence of an instance of p_j that originates from $(x,y)_i$. Given p_j and $(x,y)_i$, Degree-weighted Path Count (DWPC) [52], measures the prevalence of the instances of p_j in the neighbourhood of $(x,y)_i$. For a ground atom $a = r_0(e_0, e_1)$, we define the degree of e_0 w.r.t a as:

$$d_a(e_0) = |\{y|r_0(e_0,y) \in \mathscr{T}\}| \tag{5.13}$$

and correspondingly, the degree of e_1 w.r.t a as:

$$d_a(e_1) = |\{x|r_0(x,e_1) \in \mathscr{T}\}| \tag{5.14}$$

These degrees reflect the numbers of the one-hop neighboring entities of e_0 and e_1, respectively, under the constraints of the hop direction and predicate type inferred from the ground atom a. Given a body grounding $b_1 = (a_0, a_1)$ with $a_0 = r_0(e_0, e_1)$ and $a_1 = r_1(e_1, e_2)$, a corresponding degree vector $D_{b_1} = \{d_{a_0}(e_0), d_{a_0}(e_1), d_{a_1}(e_1), d_{a_1}(e_2)\}$ can be generated. Now, we can define the DWPC value for a rule p_j and an entity pair $(x,y)_j$ as:

$$s(\mathscr{G}, p_j, (x,y)_i)_{dwpc} = \sum_{b \in B_{p_j}^{(x,y)_i}} \prod_{d \in D_b} d^{-w} \tag{5.15}$$

where $B_{p_j}^{(x,y)_i} = \{b_p | p \in In(p_j), h_p = (x,y)_j\}$ is a set containing all the body groundings of the instances of p_j that have the head grounding $(x,y)_j$, and a dumping factor $w \in [0,1]$ that penalizes the occurrence of high-degree entities. Given a pair of rule p_j and entity pair $(x,y)_j$, DWPC assigns to each instance of p_j originated from $(x,y)_j$ a score which measures how specific the path is. The idea about the path specificity is that the larger the number of high-degree entities occurring in the path is, it is more likely that the concept expressed by the path is general because high-degree entities often represent high-level concepts. It is expensive to compute the exact DWPC values as it needs to retrieve all instantiation of a rule for a given pair. For better scalability, we compute approximate DWPC values by employing random walkers to extract only subsets of rule groundings.

5.3 Experiments

In this section, we show the experiment results of Ranta and other state-of-the-art graph reasoning systems on two large-scale biomedical KGs for drug repurposing. We first demonstrate the superiority of Ranta in predictive performance over existing works, then conduct a series of internal analysis to understand the factors affecting Ranta's performance. Eventually, a case study where Ranta suggests known drugs with potential indication for PAH is provided.

5.3.1 Datasets

Many KGs that fuse multiple types of data from various sources have been made publicly available for drug repurposing, including the recently published DRKG [57] and OpenBioLink [15]. In this work, we test the proposed and state-of-the-art symbolic systems on two large-scale biomedical KGs, the Hetionet [52] and Repotrial. Their statistics are shown in Table.5.1.

Dataset	#Entity	#Entity Types	#Triples	#Triple Types	$\|I^+\|$	$\|I^-\|$	#Drugs	#Diseases
Hetionet	47k	11	2.25m	24	755	3.02k	1.5k	137
Repotrial	350k	7	14.21m	13	3.9k	1.05k	13.3k	24.12k

Table 5.1 Statistics of Hetionet and Repotrial.

Hetionet combines biological, clinical, pharmacological information from 29 public resources to model drug efficacy. It contains 1552 small molecule drugs and 137 diseases among which 755 treatments are identified. The target predicate for Hetionet is "TREATS_CtD". Repotrial is a private KG that is constructed under the REPO-TRIAL project aiming to explore novel strategies for in silico drug repurposing. Repotrial compiles 13300 drugs, including 11268 small molecule drugs and 2032 biotech drugs, and 24120 diseases in the KG. 3906 known treatment relationships are recorded where the treatment predicate is represented as "HasIndication". Hetionet employs a gene-centric data modeling strategy where 16 of 24 predicates involve the participation of gene entities, which results in that the majority of mined regularities are explained in terms of interactions with gene. In contrast to how Hetionet excludes the inclusion of protein entities and encodes the influence of protein-related interactions in high-level relations such as the binding association between drugs and genes, Repotrial pivots towards modeling around protein interactions, which leads to the fact that 7 of 13 predicates are protein-related, such as protein-protein interaction, gene-protein expression and drug-protein target. True negative examples are not recorded in either KG. Different to the negative tuning examples generated in a rule model tuner, true negative examples, together with true positive ones, are utilized to train, validate and test the logistic regression model, thus must be prepared beforehand. For Hetionet, we follow the strategy used in the original paper where 3020 unknown drug-disease pairs are randomly selected as the negatives. For Repotrial, a set of 1052 true negative pairs are generated by consulting a gold standard database, namely repoDB [16], that contains both positive and negative drug-disease associations. We consider a possible drug-disease candidate in Repotrial negative if a terminated, withdrawn or suspended trial that examines whether the drug has indication for the disease can be found in repoDB.

5.3.2 Experiment Setup

Ranta is implemented in Java on top of the Neo4j[2] graph database. We employ the implementation of GPFL augmented with RHF (introduced in Chapter.4) as the base rule learner. We use the stratified nested cross-validation strategy [61] to evaluate Ranta. Specifically, given a set I that contains both known positive and negative examples, an external stratified 5-fold cross-validation is applied to I to create 5 example batches where for each batch, a fold is held out as the test set I_t only used for the evaluation of the logistic regression model, and the other four folds I_r are used for rule learning and rule model tuning. In rule learning, we set $I_l^+ = I_r^+$ and further split I_l^+ into training and validation sets in a 6:4 ratio where the validation set is used by GPFL to mitigate the overfitting issue. In rule model tuning, we set $I_u^+ = I_l^+$ and $I_u^- \in \{I_{lcwa}^-, I_{rudik}^-\}$. In matrix

[2]https://github.com/neo4j/neo4j

	Hetionet		Repotrial	
System	AUPRC	AUROC	AUPRC	AUROC
RLvLR [90]	.711	.852	.952	.877
AMIE+ [39]	.733	.865	.959	.891
AnyBURL [75]	.807	.932	.982	.933
GPFL+IR1	.766	.894	.968	.902
GPFL+IR2	.818	.941	.982	.937
Rephetio [52]	.867	.948	.981	.931
Ranta+IR1+Bin	.895	.956	.981	.937
Ranta+IR1+DWPC	.897	.959	.982	.941
Ranta+IR2+Bin	.909	.962	.981	.943
Ranta+IR2+DWPC	**.932**	**.977**	**.987**	**.957**

Table 5.2 Predictive performance of various systems on Hetionet and Repotrial, where the top section contains systems utilizing logic inference, and the bottom section shows feature engineering based systems. "+IRn" means the system includes IRs of maximal length n. "+Bin" and "+DWPC" indicate that feature values are binary and DWPC values, respectively. The best results are marked bold.

building, given a set of rule features R, we generate a matrix $M \in \mathbb{R}^{|I|\times|R|}$ for all known examples. In the training of a logistic regression model, an internal stratified 3-fold cross-validation is applied to the rows representing examples in I_r for model selection. Eventually, the average of system performances is reported. For the evaluation of other symbolic systems, we use a 5-fold cross-validation strategy as they do not require the tuning of hyper-parameters. In the evaluation of Ranta, we use the following common settings: the length of ARs is set to 4; the number of random walkers to 600; the offset factor in Equation.5.5 to 5; α and β in Equation.5.11 to 0.7 and 0.3, respectively, and the dumping factor w in Equation.5.15 to 0.4. On Repotrial, we set the loss threshold th to 0.07 and runtime constraint ti to 10800s, and on Hetionet, th to 0.05 and ti to 18000s. We conducted all experiments, with Ranta and other systems, on a server that has a 32 cores Intel Xeon CPU and 128GB RAM. When multi-threading is possible, we run experiments with 16 threads.

5.3.3 Evaluation Protocol

Instead of employing popular information retrieval metrics used in rule learning works [90, 75, 45], such as hits@n and Mean Reciprocal Rank (MRR) [76], a large amount of drug repurposing systems [138, 142, 73] evaluate system performance in AUROC (Area under the Receiver Operator Characteristic curves) [95]. AUROC shows how true positive rate (TPR) varies with false positive rate (FPR). However, as argued by Davis and Goadrich [24], given that TPR and FPR are computed by having the total number of positives and negatives as the denominator, respectively, AUROC is sensitive to data imbalance, thus can be overly optimistic about the system performance. Considering there exists imbalance between the numbers of positive and negative examples in Hetionet and Repotrial, in addition to AUROC, we also use AUPRC (Area

		Hetionet			Repotrial		
System	*rw*	Runtime	AUPRC	AUROC	Runtime	AUPRC	AUROC
Ranta	200	448s	.822	.935	146s	.979	.929
	400	2589s	.848	.938	617s	.980	.931
	600	6774s	.865	.944	991s	.981	.932
Rephetio	-	73376s	.867	.948	12480s	.981	.931

Table 5.3 Experiment results of Ranta with different number of random walkers (*rw*) on Hetionet and Repotrial. In this setting, Ranta only contains AR features, thus the runtime of Ranta only includes the time spent on rule generation and matrix building.

under Precision-Recall curves) for evaluation. AUPRC shows the changes in precision with varying recall. As the value of precision is independent of the total numbers of positive and negative examples, AUPRC is less sensitive to the skewness in data.

5.3.4 Predictive Performance

Stated in Table.5.2 is the predictive performance of state-of-the-art symbolic systems and Ranta in different settings. All systems are required to mine rules with maximal length of 4. RLvLR and AMIE+ were executed in default setting where only ARs are produced, and we ran AnyBURL for 10000s. For systems using logic to infer new facts, it is not apparent what strategy to adopt to assign a single confidence score to an inferred entity pair because an entity pair can be suggested by multiple rules. Consider we have a set of rules P and an inferred pair (x,y), the subset $P' = \{p \in P | (x,y) \in H_p\}$ contains all rules suggesting (x,y). We denote the quality measure of rule p by $score(p)$. A strategy is needed to score (x,y) based on $score(p)$ with $p \in P'$. To accommodate the difference in the choice of rule quality measures over various systems, where RLvLR uses an embedding-based measure; AMIE+ uses a LCWA-based one and AnyBURL and GPFL employ $smc(p)$, we utilize the Noise-or aggregation [39] to produce a single confidence value for each inferred pair for the computation of AUROC and AUPRC. Formally, given a pair (x,y) and its corresponding P', its confidence score is defined as:

$$conf((x,y)) = \prod_{p \in P'} score(p) \tag{5.16}$$

Then, by ranking inferred examples by confidence, the AUPRC and AUROC results of logic inference based systems are computed.

As observed in Table.5.2, the feature engineering based systems significantly outperform the logic inference based ones with the exception that AnyBURL and GPFL show slightly better performance than Rephetio on Repotrial. Evidently, Ranta in all settings on both Hetionet and Repotrial outperform Rephetio, which supports our assumption that the introduction of IRs can further improve system performance. For instance, Ranta with hop-2 IRs and DWPC outperforms Rephetio by 7.5% and 3.1% in AUPRC and AUROC on Hetionet. The observation

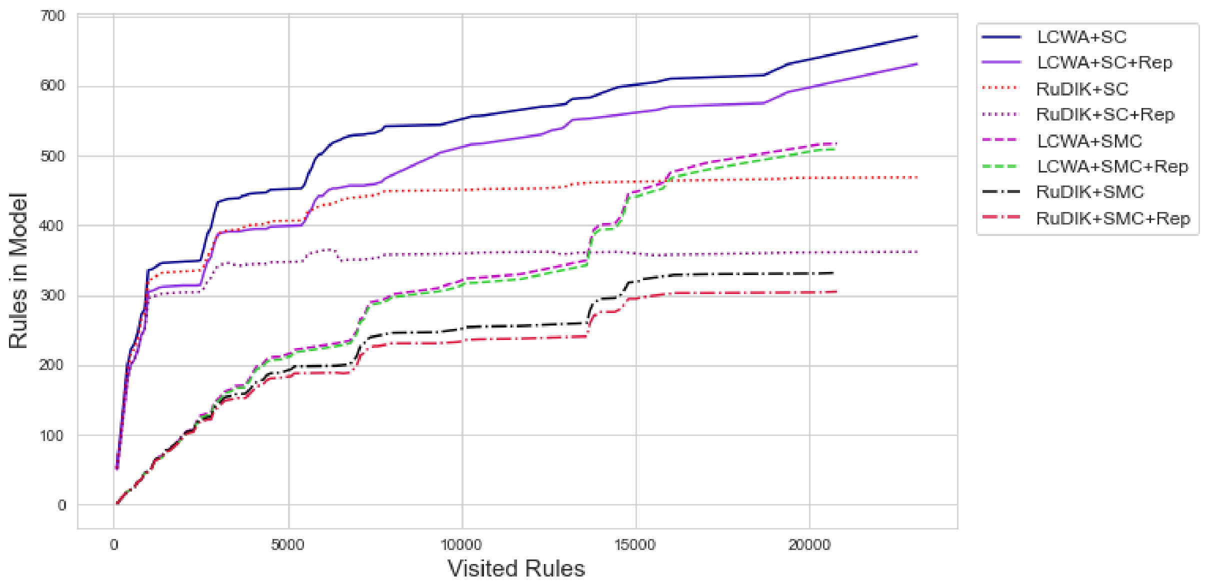


Fig. 5.3 Number of rules visited before reaching convergence in different settings on Repotrial with Ranta.

that systems with 2-hop IRs show better performances than ones with one-hop IRs reconfirms the assumption proposed by Himmelstein et al. [52], that is to better model drug efficacy, long rules that represent complex biomedical concepts are needed. Another interesting observation is that systems with DWPC as feature values show better performance than ones with binary values, which contradicts the popular belief [41] that specificity-based values have no apparent benefits to predictive performance compared to binary values.

5.3.5 Ablation Study

In this section, we show a series of experiment results to demonstrate the impact of the choices of the number of random walkers, the use of the replacement mechanism, the negative sampling strategy and the quality measure on predictive performance, runtime and the convergence of rule model tuning.

Random Walkers

As discussed in Section.5.2, the choice of the number of random walkers, denoted by *rw*, has a considerable impact on predictive performance and runtime. This is because random walkers are used not only for the generation of rules, but also the estimation of DWPC values. In Table.5.3, we observe that the runtime and predictive performance on Hetionet vary drastically with different *rw*, whereas on Repotrial, the improvements with increasing *rw* are more subtle. The reason for capping *rw* at 600 is that we observed the changes in runtime and predictive performance become stable on Hetionet with *rw* at around 600, and on Repotrial, the system performance stabilized with *rw* at around 400. On the surface, it is counter-intuitive that Repotrial requires less time to run than Hetionet given that Repotrial has a much larger collection of entities and triples. As measured in experiments, because Hetionet has more predicates than Repotrial,

Settings	Replacement	#IRs	Runtime	AUROC
LCWA+SC	Yes	631	9437s	.953
	No	671	6921s	.949
LCWA+SMC	Yes	509	8815s	.956
	No	517	6432s	.951
RuDIK+SC	Yes	362	5810s	.952
	No	469	5644s	.948
RuDIK+SMC	Yes	**305**	5532s	**.957**
	No	332	**5319s**	.953

Table 5.4 Experiment results of Ranta in different rule model tuning settings on Repotrial. SC and SMC stand for $sc(p)$ and $smc(p)$, respectively. #IRs means the number of IRs in the generated rule model, and the runtime indicates the time used to finish the rule model tuning. Best results are marked bold.

Hetionet had 1072 AR features which is much larger than Repotrial's 139 AR features. When we factored in the number of examples, Hetionet needed to fill 4.04 million DWPC values in the matrix, whereas Repotrial only needed to compute 687 thousands. Therefore, Hetionet required much more time to run than Repotrial. Moreover, because the DWPC values with Ranta are estimated based on the paths extracted by random walkers, the smaller *rw* is, less time is required to compute DWPC values. Thus, we observe that although the number of AR features is similar between systems in different *rw* settings, runtime differs significantly. Impressively, compared to Rephetio that computes the exact DWPC values, Ranta with estimated DWPC values take much less time to build models that perform slightly worse than Rephetio, which implies that properly estimated DWPC values are enough to make good predictions.

Rule Model Tuning

Table.5.4 shows the experiment results of Ranta with a rule model tuner configured in different settings. One observation we make is that, in comparison to LCWA strategy, the use of RuDIK strategy for negative sampling significantly reduces the runtime needed for tuning the rule model without compromising the predictive performance. This is because RuDIK strategy requires the entities in negative examples to be semantically related, thus results in a much smaller set of negative examples than that produced by LCWA. In experiments, we had $|I^-_{lcwa}|$ in the size of millions and $|I^-_{rudik}|$ in thousands on Repotrial. Smaller set of negative examples means the operations, such as union and interaction, on the set become cheaper to execute, thus the decrease in runtime. We also observe that rule models with less IRs often show relatively better performances. We attribute this to the removal of redundant and noisy rules because given the different sizes of rule models over experiments with the same negative sampling strategies in Table.5.4, they share the same loss value, which implies that they cover the same set of positive and negative examples. The activation of the replacement mechanism further removes and replaces redundant and noisy rules, which leads to the improvement in performance. As

Type	Rule	Coef
CAR	$Indication(X,Y) \leftarrow Indication(X,V_0), Indication(V_1,V_0), Indication(V_1,Y)$	1.082
CAR	$Indication(X,Y) \leftarrow Contraindication(X,V_0), Contraindication(V_1,V_0), Indication(V_1,Y)$	0.806
CAR	$Indication(X,Y) \leftarrow Indication(X,V_0), IsDisorder(V_0,V_1), IsDisorder(Y,V_1)$	0.684
BAR	$Indication(X, breast\ carcinoma) \leftarrow Indication(X,V_0), Indication(V_0, neuroblastoma)$	0.561
BAR	$Indication(X, rheumatoid\ arthritis) \leftarrow Indication(X,V_0), IsDisorder(V_0, AMD)$	0.558
BAR	$Indication(X, acute\ myeloid\ leukemia) \leftarrow Indication(X,V_0), IsDisorder(V_0, myeloid\ leukemia)$	0.532
CAR	$Indication(X,Y) \leftarrow HasTarget(X,V_0), HasTarget(V_1,V_0), Indication(V_1,Y)$	0.525
BAR	$Indication(X,Y) \leftarrow Contraindication(X,V_0), Comorbid(V_1,V_0), Contraindication(V_2,V_1), Indication(V_2,Y)$	0.516
BAR	$Indication(Doxorubicin,Y) \leftarrow Indication(V_0,Y), HasTarget(V_0, TERT\_HUMAN)$	0.514
CAR	$Indication(X,Y) \leftarrow Similar(X,V_0), Similar(V_0,V_1), Similar(V_2,V_1), Indication(V_2,Y)$	0.509

Table 5.5 Top-ranked rule features in the best performing Ranta model on Repotrial. AMD stands for autoimmune musculoskeletal disease.

illustrated in Figure.5.3, we observe that the combination of RuDIK and $smc(p)$ drastically reduces the size of rule model, and the number of visited rules before convergence is dependent only on the choice of quality measures.

Interpretability

One of the most important advantages of Ranta and other symbolic systems is that the learned model is interpretable. Illustrated in Table.5.5 is a set of top rule features ranked by coefficients of the trained logistic regression model. Each feature represents a pattern that explains a statistically important aspect of the drug efficacy model underlying Repotrial. For instance, the first rule states that if drugs X and V_1 have shared indications, then the diseases Y that can be treated by V_1 can also be treated by X. This learned pattern expresses, in logic form, one of the most commonly adopted guilt-by-association strategy stating that similar drugs may treat common diseases. In comparison to Rephetio, Ranta also includes IRs as features. For instance the rule:

$$Indication(X, breast\ cancer) \leftarrow Indication(X,V_0), Indication(V_0, neuroblastoma)$$

is a BAR and states that drugs X for neuroblastoma also have indication for breast cancer. Neuroblastoma is one of the most common childhood tumours that arise from nerve cells in either the chest or the abdomen. A recent study [59] suggests that the PARP inhibitors conventionally used for ovarian and breast cancers are effective in treating neuroblastoma, which confirms the potential correlation between breast caners and neuroblastoma as suggested by the proposed BAR.

5.3.6 Case Study

To further demonstrate that Ranta can make reasonable repurposing suggestions, we conducted a case study that aims to identify top repurposing opportunities for Pulmonary Arterial Hypertension (PAH). We first created a set of candidate drug-disease pairs where the drugs are not already in a known therapeutic relationship with PAH in the KG. Then, we built a feature matrix for the candidate pairs and passed the matrix to an existing model trained by Ranta with IRs to assign an estimated probability to each pair. Table.5.6 lists the top repurposing opportunities

Rank	Status	Drug
1	Unknown	Vardenafil
2	Unknown	Dinoprostone
3	In-trial	Udenafil
4	In-trial	Tadalafil
5	Unknown	Mirodenafil
6	Unknown	Dinoprost Tromethamine
7	In-trial	Carvedilol
8	Unknown	Phenylpropanolamine
9	Unknown	Phenylephrine
10	Unknown	Travoprost

Table 5.6 Top-ranked repurposed drugs for PAH. The in-trial status is retrieved from the ClinicalTrials.gov database. Unknown indicates the existence of the drug's indication for PAH is not recorded in ClinicalTrials.gov and the KG.

Type	Rule	Coef	Value
CAR	$Indication(X,Y) \leftarrow Indication(X,V_0), Indication(V_1,V_0), Indication(V_1,Y)$	1.082	0.041
CAR	$Indication(X,Y) \leftarrow Contraindication(X,V_0), Contraindication(V_1,V_0), Indication(V_1,Y)$	0.806	0.052
CAR	$Indication(X,Y) \leftarrow HasTarget(X,V_0), HasTarget(V_1,V_0), Indication(V_1,Y)$	0.525	0.072
CAR	$Indication(X,Y) \leftarrow SimilarMolecule(X,V_0), SimilarMolecule(V_1,V_0), Indication(V_1,Y)$	0.279	0.076
BAR	$Indication(X,'PAH') \leftarrow Indication(X,V_0), Indication(slidenafil,V_0)$	0.136	0.143

Table 5.7 Evidence of repurposing vardenafil for PAH. Value is the DWPC value for the rule feature and $(vardenafil, pah)$ pair.

ranked by probability. 3 of the top 10 repurposed drugs are already in trial according to the ClinicalTrials.gov database. Moreover, the effectiveness of vardenafil and mirodenafil for PAH is supported by literature [58, 124]. In Table.5.7, we demonstrate Ranta's capability of supporting predictions with IRs, which explains specific evidences that can not be expressed by ARs. For instance, the rule:

$$Indication(X,PAH) \leftarrow Indication(X,V_0), Indication(slidenafil,V_0)$$

states that if a drug X has shared indications with slidenafil [7], an approved PDE-5 inhibitor used for the treatment of PAH, then X treats PAH. Through this case study, we show that Ranta is capable of making reasonable suggestions supported with interpretable evidences.

5.4 Conclusion

In this chapter, we have introduced the importance of drug repurposing and the necessity of developing interpretable knowledge graph reasoning systems that can be applied on biomedical KGs with information integrated from different sources for computational drug repurposing. To such end, we have developed a feature engineering based system that converts examples and the logical rules mined from KGs into a feature matrix and trains a logistic regression model with the matrix to make repurposing predictions. Unlike existing works that either employ features of simple path types or binary feature value, we use an optimization approach to identify a minimal set of IRs as additional features to complement simple features. In addition, we use a degree-based feature value to measure the plausibility of features. Together, the system has achieved impressive improvements in execution time, prediction accuracy and interpretability. Specifically, through the use of random walkers and the implementation of a bi-directional graph traverser, Ranta runs more than 10 times faster than Rephetio without compromising predictive performance on certain tasks. Over experiments on two biomedical KGs, we confirmed our assumption that the introduction of informative IRs could further improve the predictive performance of feature engineering based symbolic methods, where Ranta with IRs significantly outperforms other state-of-the-art systems. The interpretability of Ranta is exhibited through the exploration of the top-ranked logical features in a trained model, and a case study where we repurposed drugs for PAH. A large number of the top ranked drugs for PAH are either in trial or known to be effective to PAH.

Ranta shows what well-designed biomedical knowledge graphs can do when combined with a strong algorithm. Making drug repurposing predictions is merely one application of Ranta. Ranta can also be used to predict what a new drug can treat or correct errors in biomedical knowledge bases. The interpretability of Ranta is valuable when practitioners attempt to understand the reasoning and evidence behind machine's predictions. However, feature engineering based methods work less well on modeling molecule structure to enable virtual screening [127]. Graph neural networks (GNNs) have been successfully employed for in-silico drug discovery [135, 132]. The unification of symbolic methods and GNNs for drug discovery is a promising research direction where both good generalization ability and interpretability can be achieved.

Chapter 6

Conclusion

In this thesis, we have introduced three contributions that aim to progress the research of the logical rule learning systems operating on Knowledge Graphs (KGs). First, we proposed GPFL, a novel probabilistic rule learning system utilizing a two-stage rule generation mechanism to optimize the mining of instantiated rules. Starting from the problems with AnyBURL, where rule evaluation is the bottleneck of system's scalability and the saturation-based higher-order function that controls the learning procedure prevents the system to mine complex rules, we utilized a novel two-stage rule learning mechanism to solve the above mentioned problems. Through experiments, we showed that GPFL significantly reduces the runtime on evaluating instantiated rules, discovers much more high-quality rules than existing works and performs competitively on knowledge graph completion task compared to existing methods. As the quality of instantiated rules is often decided by a much smaller group of groundings than that of abstract rules, instantiated rules are more prone to be overfitting. Based on this point, we conducted a series of experiments with GPFL to demonstrate the prevalence of overfitting instantiated rules and their adverse impacts on systems' performance where we provide a simple validation method to mitigate the issue.

Second, we proposed RHF, a generic framework that leverages a collection of novel subsumption frameworks to build proper rule hierarchies from a given set of rules. By defining the SA-subsumption which reduces the complexity of classic OI-subsumption from exponential to linear, we developed two derived subsumption frameworks, namely A-subsumption and I-subsumption, to enable the efficient construction of proper rule hierarchies. Along with Hierarchical Pruning Methods (HPMs), we applied RHF to GPFL to evaluate the effectiveness of the proposed design. With experiment results on four benchmark KGs, we demonstrated that the application of RHF and HPMs effectively removes redundant and irrelevant rules, which results in significant reductions in the runtime and the number of learned rules, without compromising predictive performance. In comparison to recent works [63, 54] that aim to develop pruning technologies for top-down rule learners, RHF aims to provide HPMs support to bottom-up rule learners that overtake the top-down approaches in recent years.

Finally, we designed Ranta, a novel drug repurposing system that converts examples and the logical rules extracted from biomedical KGs into a feature matrix and trains a logistic regression model with the matrix to model drug efficacy and make drug repurposing inferences. Through extensive experiments on two large-scale biomedical KGs, we demonstrated that Ranta largely outperforms existing methods and executes much faster than a state-of-the-art system. The case study where we ask Ranta to repurpose drugs for Pulmonary Arterial Hypertension showed that Ranta is capable of making reasonable suggestions and providing interpretable evidence which is often impossible for sub-symbolic methods that operate as a black-box.

As discussed in previous chapters, one of the main reasons that drives us to research rule learning systems is their ability to make interpretable inferences. The interpretability of Machine Learning models has been a hot research topic in recent years [29, 79] because many real-world scenarios require the prediction results to be accompanied with human-understandable explanations, whereas most of the Deep Learning models have trouble doing so. For learning from multi-relational data such as KGs and Knowledge Bases (KBs) [31], the feature engineering strategy where features, such as sub-graphs and logical rules, are extracted from the multi-relational data and used to train a conventional interpretable model is still one of the widely employed methods to approach the interpretable multi-relational machine learning problem. Ranta is one example of such systems. In contrast, Statistical Relational Learning (SRL) [60] methods, such as MLN [101] and RDN [85], are often too computationally expensive to be applied to real-world tasks. The dependence on the externally generated rules makes the feature engineering based methods subject to the lack of robustness of the conventional rule learning systems. Thus, the learning results are often not optimal. The neural-symbolic methods [64] sprouted in recent years is a rising line of research which is promising to address the lack of robustness issue. In general, a neural-symbolic system uses numerical optimization to simultaneously build the structure of and assign the confidence score to rules. For instance, Neural LP [137] and DRUM [104] are neural-symbolic rule learners based on the differentiable logic formulation TensorLog [22] where the traversals over a KG are formulated as sequences of matrix multiplications, and logical inferences are compiled into sequences of differentiable operations on matrices. Currently, most of the neural-symbolic works can only generate abstract rules due to the limitations on architecture design.

In addition to interpretability, rules are often used as external knowledge or additional information to augment Distributed Representation (DR) systems. For instance, Wang et al. [130] proposed to generate potentially missing entity pairs utilizing embeddings and then use externally generated rules as constraints to filter out inconsistent pair candidates. Moreover, recent works, including KALE [46] and RUGE [47], manage to use a joint model to embed the components in KGs and external rules simultaneously. Most of these works only use abstract rules for augmentation because the effect of instantiated rules is not well-understood. Meilicke et al. [75] argued that our understanding of the correlation between the choice of the types of rules and the systems' predictive performance is still in the early stage. Therefore, it may be necessary to explore and experiment more on rule learning before investigating integration works. Consequently, we propose two future work directions: 1.) designing a unified analytics framework for rule learning systems to understand what types of rules contribute the most to the predictive performance under various conditions and why, and 2.) developing efficient neural-symbolic systems that produce instantiated rules.

References

[1] Abedjan, Z. and Naumann, F. (2013). Improving RDF data through association rule mining. *Datenbank-Spektrum*, 13(2):111–120.

[2] Ahmadi, N., Huynh, V. P., Meduri, V., Ortona, S., and Papotti, P. (2020). Mining Expressive Rules in Knowledge Graphs. *Journal of Data and Information Quality*, 12(2).

[3] Auer, S., Bizer, C., Kobilarov, G., Lehman, J., Cyganiak, R., and Ives, Z. (2007a). DBedpia: A Nucleus for a Web od Open Data. *Semantic Web Journal*, pages 722–735.

[4] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007b). Dbpedia: A nucleus for a web of open data. In *The Semantic Web*, pages 722–735. Springer.

[5] Balazevic, I., Allen, C., and Hospedales, T. (2019). TuckER: Tensor Factorization for Knowledge Graph Completion. In *Conference on Empirical Methods in Natural Language Processing*, pages 5184–5193.

[6] Barati, M., Bai, Q., and Liu, Q. (2016). SWARM: An approach for mining semantic association rules from semantic web data. In *Trends in Artificial Intelligence*, pages 30–43. Springer International Publishing.

[7] Barnett, C. F. and Machado, R. F. (2006). Sildenafil in the treatment of pulmonary hypertension. *Vasc. Health Risk Manag.*, 2(4):411–422.

[8] Blockeel, H., Dehaspe, L., Demoen, B., Janssens, G., Ramon, J., and Vandecasteele, H. (2002). Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research*, 16:135–166.

[9] Bodenreider, O. (2004). The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

[10] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008a). Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM Special Interest Group on Management of Data*, pages 1247–1250.

[11] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008b). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

[12] Bordes, A., Chopra, S., and Weston, J. (2014). Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Stroudsburg, PA, USA. Association for Computational Linguistics.

[13] Bordes, A., Usunier, N., Weston, J., and Yakhnenko, O. (2013). Translating Embeddings for Modeling Multi-Relational Data. In *Conference on Neural Information Processing Systems*, pages 2787–2795.

[14] Breckenridge, A. and Jacob, R. (2019). Overcoming the legal and regulatory barriers to drug repurposing. *Nat. Rev. Drug Discov.*, 18(1):1–2.

[15] Breit, A., Ott, S., Agibetov, A., and Samwald, M. (2020). OpenBioLink: A benchmarking framework for large-scale biomedical link prediction. *Bioinformatics*, 30:2–3.

[16] Brown, A. S. and Patel, C. J. (2017). A standard database for drug repositioning. *Scientific data*, 4(1):1–7.

[17] Bytyçi, E., Ahmedi, L., and Kurti, A. (2016). Association rule mining with context ontologies: An application to mobile sensing of water quality. In *Metadata and Semantics Research*, pages 67–78. Springer International Publishing.

[18] Cao, X. (2020). COVID-19: immunopathology and its implications for therapy. *Nat. Rev. Immunol.*, 20(5):269–270.

[19] Chen, X., Chen, M., Shi, W., Sun, Y., and Zaniolo, C. (2018). Embedding uncertain knowledge graphs. *arXiv e-prints*, page arXiv:1811.10667.

[20] Chen, X., Jia, S., and Xiang, Y. (2020). A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141.

[21] Chen, Y., Wang, D. Z., and Goldberg, S. (2016). ScaLeKB: scalable learning and inference over large knowledge bases. *VLDB Journal*, 25(6):893–918.

[22] Cohen, W. W. (2020). TensorLog : A Probabilistic Database Implemented Using Deep-Learning Infrastructure. *Journal of Artificial Intelligence Research*, 67:285–325.

[23] Darari, F., Nutt, W., Pirrò, G., and Razniewski, S. (2018). Completeness management for RDF data sources. *ACM trans. web*, 12(3):1–53.

[24] Davis, J. and Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 233–240, New York, NY, USA. Association for Computing Machinery.

[25] De Raedt, L. (2008). *Logical and relational learning*. Springer Science & Business Media.

[26] Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. (2018). Convolutional 2D Knowledge Graph Embeddings. In *Conference on Artificial Intelligence (AAAI)*, pages 1811–1818.

[27] DiMasi, J. A., Grabowski, H. G., and Hansen, R. W. (2016). Innovation in the pharmaceutical industry: New estimates of R&D costs. *J. Health Econ.*, 47:20–33.

[28] Dimitrakis, E., Sgontzos, K., and Tzitzikas, Y. (2020). A survey on question answering systems over linked data and documents. *J. Intell. Inf. Syst.*, 55(2):233–259.

[29] Du, M., Liu, N., and Hu, X. (2019). Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77.

[30] Duboc, A. L., Paes, A., and Zaverucha, G. (2009). Using the bottom clause and mode declarations in FOL theory revision from examples. *Machine Learning*, 76(1):73–107.

[31] Džeroski, S. (2003). Multi-relational data mining. *ACM SIGKDD Explorations Newsletter*, 5:1.

[32] Ernst, P., Meng, C., Siu, A., and Weikum, G. (2014). KnowLife: A knowledge graph for health and life sciences. In *2014 IEEE 30th International Conference on Data Engineering*, pages 1254–1257. ieeexplore.ieee.org.

[33] Fensel, D., Şimşek, U., Angele, K., Huaman, E., Kärle, E., Panasiuk, O., Toma, I., Umbrich, J., and Wahler, A. (2020). Knowledge Graphs. In *Arxiv*.

[34] Ferilli, S., Fanizzi, N., Di Mauro, N., and Basile, T. M. A. (2002). Efficient theta-Subsumption under Object Identity. In *International Conference of the Italian Association for Artificial Intelligence (AI*IA)*.

[35] Fetro, C. and Scherman, D. (2020). Drug repurposing in rare diseases: Myths and reality. *Therapie*, 75(2):157–160.

[36] França, M. V., Zaverucha, G., and D'Avila Garcez, A. S. (2014). Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine Learning*, 94(1):81–104.

[37] Gad-Elrab, M. H., Stepanova, D., Urbani, J., and Weikum, G. (2016). Exception-enriched rule learning from knowledge graphs. In *International Semantic Web Conference (ISWC)*, volume 9981 LNCS, pages 234–251.

[38] Gadkar, K., Kirouac, D., Parrott, N., and Ramanujan, S. (2016). Quantitative systems pharmacology: a promising approach for translational pharmacology. *Drug Discov. Today Technol.*, 21-22:57–65.

[39] Galárraga, L., Teflioudi, C., Hose, K., and Suchanek, F. M. (2015). Fast rule mining in ontological knowledge bases with AMIE+. *VLDB Journal*, 24(6):707–730.

[40] Gallicchio, C. and Micheli, A. (2010). Graph echo state networks. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. ieeexplore.ieee.org.

[41] Gardner, M. and Mitchell, T. (2015). Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction. In *Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498.

[42] Ge, Y., Tian, T., Huang, S., Wan, F., Li, J., Li, S., Yang, H., and others (2020). A data-driven drug repositioning framework discovered a potential therapeutic agent targeting COVID-19. *bioRxiv*.

[43] Gong, F., Ma, Y., Gong, W., Li, X., Li, C., and Yuan, X. (2018). Neo4j graph database realizes efficient storage performance of oilfield ontology. *PLoS One*, 13(11):e0207595.

[44] Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2. ieeexplore.ieee.org.

[45] Gu, Y., Guan, Y., and Missier, P. (2020). Towards learning instantiated logical rules from knowledge graphs. *arXiv: Artificial Intelligence*.

[46] Guo, S., Wang, Q., Wang, L., Wang, B., and Guo, L. (2016). Jointly embedding knowledge graphs and logical rules. In *Conference on Empirical Methods in Natural Language Processing*, pages 192–202.

[47] Guo, S., Wang, Q., Wang, L., Wang, B., and Guo, L. (2018). Knowledge Graph Embedding with Iterative Guidance from Soft Rules. In *Conference on Artificial Intelligence (AAAI)*, pages 4816–4823.

[48] Gysi, D. M., do Valle, Í., Zitnik, M., Ameli, A., Gan, X., Varol, O., Ghiassian, S. D., Patten, J. J., Davey, R. A., Loscalzo, J., and Barabási, A.-L. (2021). Network medicine framework for identifying drug-repurposing opportunities for COVID-19. *Proc. Natl. Acad. Sci. U. S. A.*, 118(19).

[49] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Conference on Neural Information Processing Systems*.

[50] Hefti, F. F. (2008). Requirements for a lead compound to become a clinical candidate. *BMC Neurosci.*, 9 Suppl 3:S7.

[51] Hernandez, J. J., Pryszlak, M., Smith, L., Yanchus, C., Kurji, N., Shahani, V. M., and Molinski, S. V. (2017). Giving drugs a second chance: Overcoming regulatory and financial hurdles in repurposing approved drugs as cancer therapeutics. *Front. Oncol.*, 7:273.

[52] Himmelstein, D. S., Lizee, A., Hessler, C., Brueggeman, L., Chen L, S., Hadley, D., Green, A., Khankhanian, P., and Baranzini, S. E. (2017). Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLIFE*, 6:1–35.

[53] Hitzler, P., Bianchi, F., Ebrahimi, M., and Sarker, M. K. (2020). Neural-symbolic integration and the semantic web. *Semantic Web*, 11(1):3–11.

[54] Ho, V. T., Stepanova, D., Gad-Elrab, M. H., Kharlamov, E., and Weikum, G. (2018). Rule learning from knowledge graphs guided by embedding models. In *International Semantic Web Conference (ISWC)*, pages 72–90.

[55] Hodos, R. A., Kidd, B. A., Shameer, K., Readhead, B. P., and Dudley, J. T. (2016). In silico methods for drug repurposing and pharmacology. *Wiley Interdiscip. Rev. Syst. Biol. Med.*, 8(3):186–210.

[56] Hoffart, J., Suchanek, F. M., Berberich, K., and Weikum, G. (2013). YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194:28–61.

[57] Ioannidis, V. N., Song, X., Manchanda, S., Li, M., Pan, X., Zheng, D., Ning, X., Zeng, X., and Karypis, G. (2020). Drkg - drug repurposing knowledge graph for covid-19. https://github.com/gnn4dr/DRKG/.

[58] Jing, Z.-C., Yu, Z.-X., Shen, J.-Y., Wu, B.-X., Xu, K.-F., Zhu, X.-Y., Pan, L., Zhang, Z.-L., Liu, X.-Q., Zhang, Y.-S., Jiang, X., Galiè, N., and Efficacy and Safety of Vardenafil in the Treatment of Pulmonary Arterial Hypertension (EVALUATION) Study Group (2011). Vardenafil in pulmonary arterial hypertension: a randomized, double-blind, placebo-controlled study. *Am. J. Respir. Crit. Care Med.*, 183(12):1723–1729.

[59] King, D., Li, X. D., Almeida, G. S., Kwok, C., Gravells, P., Harrison, D., Burke, S., Hallsworth, A., Jamin, Y., George, S., Robinson, S. P., Lord, C. J., Poon, E., Yeomanson, D., Chesler, L., and Bryant, H. E. (2020). MYCN expression induces replication stress and sensitivity to PARP inhibition in neuroblastoma. *Oncotarget*, 11(23):2141–2159.

[60] Koller, D., Friedman, N., Džeroski, S., Sutton, C., McCallum, A., Pfeffer, A., Abbeel, P., Wong, M.-F., Heckerman, D., Meek, C., et al. (2007). *Introduction to statistical relational learning*. MIT press.

[61] Krstajic, D., Buturovic, L. J., Leahy, D. E., and Thomas, S. (2014). Cross-validation pitfalls when selecting and assessing regression and classification models. *J. Cheminform.*, 6(1):10.

[62] Kumar, S. (2020). COVID-19: A drug repurposing and biomarker identification by using comprehensive Gene-Disease associations through Protein-Protein interaction network analysis. *bioRxiv*.

[63] Lajus, J., Galárraga, L., and Suchanek, F. (2020). Fast and Exact Rule Mining with AMIE 3. In *European Semantic Web Conference*, pages 36–52.

[64] Lamb, L. C., Garcez, A. D., Gori, M., Prates, M. O. R., Avelar, P. H. C., and Vardi, M. Y. (2020). Graph neural networks meet neural-symbolic computing: A survey and perspective. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, California. International Joint Conferences on Artificial Intelligence Organization.

[65] Lao, N., Minkov, E., and Cohen, W. (2015). Learning Relational Features with Backward Random Walks. In *Annual Meeting of the Association for Computational Linguistics*, pages 666–675.

[66] Lao, N., Mitchell, T., and Cohen, W. W. (2011). Random Walk Inference and Learning in A Large Scale Knowledge Base. In *Conference on Empirical Methods in Natural Language Processing*, pages 529–539.

[67] Lenat, D. B. (1995). CYC: a large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):33–38.

[68] Lerer, A., Wu, L., Shen, J., Lacroix, T., Wehrstedt, L., Bose, A., and Peysakhovich, A. (2019). PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference*, Palo Alto, CA, USA.

[69] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015). Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Conference on Artificial Intelligence (AAAI)*, pages 2181–2187.

[70] Liu, J., Lu, Z., and Du, W. (2019). Combining enterprise knowledge graph and news sentiment analysis for stock price prediction. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*. 128.171.57.22.

[71] Ma, Y., Peng, H., and Cambria, E. (2018). Targeted Aspect-Based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. *Conference on Artificial Intelligence (AAAI).*

[72] Maloberti, J. and Sebag, M. (2004). Fast Theta-Subsumption with Constraint Satisfaction Algorithms. *Machine Learning*, 55(2):137–174.

[73] Martínez, V., Cano, C., and Blanco, A. (2014). ProphNet: A generic prioritization method through propagation of information. *BMC Bioinformatics*, 15(1):S5.

[74] Meilicke, C., Chekol, M. W., Fink, M., and Stuckenschmidt, H. (2020). Reinforced anytime bottom up rule learning for knowledge graph completion. *arXiv*.

[75] Meilicke, C., Chekol, M. W., Ruffinelli, D., and Stuckenschmidt, H. (2019). Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3137–3143.

[76] Meilicke, C., Fink, M., Wang, Y., Ruffinelli, D., Gemulla, R., and Stuckenschmidt, H. (2018). Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion. In *International Semantic Web Conference (ISWC)*, pages 3–20.

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

[78] Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang, R., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., and Welling, J. (2018). Never-ending learning. *Communications of the ACM*, 61(5):103–115.

[79] Molnar, C. (2020). *Interpretable Machine Learning*. Lulu.com.

[80] Muggleton, S. (1995). Inverse entailment and progol. *New Generation Computing*, 13(3-4):245–286.

[81] Muggleton, S. and de Raedt, L. (1994). Inductive Logic Programming: Theory and methods. *The Journal of Logic Programming*, 19-20:629–679.

[82] Muggleton, S., De Raedt, L., Poole, D., Bratko, I., Flach, P., Inoue, K., and Srinivasan, A. (2012). ILP turns 20: Biography and future challenges. *Machine Learning*, 86(1):3–23.

[83] Nakashole, N., Theobald, M., and Weikum, G. (2011). Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the fourth ACM international conference on Web search and data mining - WSDM '11*, New York, New York, USA. ACM Press.

[84] Nakashole, N., Weikum, G., and Suchanek, F. M. (2012). PATTY: A taxonomy of relational patterns with semantic types. *Conference on Empirical Methods in Natural Language Processing*.

[85] Neville, J. and Jensen, D. (2007). Relational dependency networks. *Journal of Machine Learning Research*, 8(Mar):653–692.

[86] Nguyen, H. L., Vu, D. T., and Jung, J. J. (2020). Knowledge graph fusion for smart systems: A survey. *Inf. Fusion*, 61:56–70.

[87] Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. (2016). A review of relational machine learning for knowledge graphs. *Proc. IEEE*, 104(1):11–33.

[88] Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning*, volume 11, pages 809–816.

[89] Nilsson, U. and Małuszyński, J. (1990). *Logic, programming and Prolog*. Wiley Chichester.

[90] Omran, P. G., Wang, K., and Wang, Z. (2018). Scalable rule learning via learning representation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2149–2155.

[91] Ortona, S., Meduri, V. V., and Papotti, P. (2018). Robust Discovery of Positive and Negative Rules in Knowledge-Bases. In *International Conference on Data Engineering (ICDE)*, pages 1168–1179. IEEE.

[92] Parvathaneni, V., Kulkarni, N. S., Muth, A., and Gupta, V. (2019). Drug repurposing: a promising tool to accelerate the drug discovery process. *Drug Discov. Today*, 24(10):2076–2085.

[93] Paulheim, H. (2016). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semant. Web*, 8(3):489–508.

[94] Philip Lewis (2020). Search is smarter with knowledge graphs and NLP. https://www.accenture.com/us-en/blogs/search-and-content-analytics-blog/enterprise-search-knowledge-graphs. Accessed: 2021-1-24.

[95] Provost, F. J., Fawcett, T., and Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 445–453, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[96] Pushpakom, S., Iorio, F., Eyers, P. A., Escott, K. J., Hopper, S., Wells, A., Doig, A., Guilliams, T., Latimer, J., McNamee, C., Norris, A., Sanseau, P., Cavalla, D., and Pirmohamed, M. (2019). Drug repurposing: progress, challenges and recommendations. *Nat. Rev. Drug Discov.*, 18(1):41–58.

[97] Qi, G., Gao, H., and Wu, T. (2017). The research advances of knowledge graph. *International Conference on Data Engineering (ICDE)*.

[98] Qi, Y., Jiang, R., Jia, Y., Li, R., and Li, A. (2018). Association analysis algorithm based on knowledge graph for SPACE-Ground integrated network. In *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, pages 222–226. ieeexplore.ieee.org.

[99] Quinlan, J. R. (1990). Learning logical definitions from relations. *Mach. Learn.*

[100] Reichert, J. M. (2003). Trends in development and approval times for new therapeutics in the united states. *Nat. Rev. Drug Discov.*, 2(9):695–702.

[101] Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine learning*, 62(1-2):107–136.

[102] Ristoski, P. and Paulheim, H. (2014). Feature Selection in Hierarchical Feature Spaces. In *International Conference on Discovery Science*, pages 288–300.

[103] Rossi, A., Barbosa, D., Firmani, D., Matinata, A., and Merialdo, P. (2021). Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data*, 15(2):1–49.

[104] Sadeghian, A., Armandpour, M., Ding, P., and Wang, D. Z. (2019). DRUM: End-To-End Differentiable Rule Mining On Knowledge Graphs. In *Conference on Neural Information Processing Systems*, pages 1–13.

[105] Scannell, J. W., Blanckley, A., Boldon, H., and Warrington, B. (2012). Diagnosing the decline in pharmaceutical R&D efficiency. *Nat. Rev. Drug Discov.*, 11(3):191–200.

[106] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Trans. Neural Netw.*, 20(1):61–80.

[107] Schlichtkrull, M., B, T. N. K., Bloem, P., Berg, R. V. D., Titov, I., and Welling, M. (2018). Modeling Relational Data with Graph Convolutional Networks. In *Extended Semantic Web Conference (ESWC)*, volume 10843, pages 593–607. Springer International Publishing.

[108] Shi, B. and Weninger, T. (2016). Discriminative predicate path mining for fact checking in knowledge graphs. *Knowledge-Based Systems*, 104:123–133.

[109] Shi, B. and Weninger, T. (2018). Open-World knowledge graph completion. *Conference on Artificial Intelligence (AAAI)*, 32(1).

[110] Shiralkar, P., Flammini, A., Menczer, F., and Ciampaglia, G. L. (2017). Finding streams in knowledge graphs to support fact checking. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 859–864. ieeexplore.ieee.org.

[111] Singh, T. U., Parida, S., Lingaraju, M. C., Kesavan, M., Kumar, D., and Singh, R. K. (2020). Drug repurposing approach to fight COVID-19. *Pharmacol. Rep.*, 72(6):1479–1508.

[112] Singhal, A. (2012). Introducing the knowledge graph: things, not strings. https://blog.google/products/search/introducing-knowledge-graph-things-not/. Accessed: 2021-1-19.

[113] Skelton, D. J., Alsobhe, A., Anastasi, E., Atallah, C., Bird, J. E., Brown, B., Didon, D., Gater, P., James, K., Lennon, Jr, D. D., McLaughlin, J., Moreland, P. E. J., Pocock, M., Whitaker, C. J., and Wipat, A. (2020). Drug repurposing prediction for COVID-19 using probabilistic networks and crowdsourced curation. *arXiv*.

[114] Socher, R., Chen, D., Manning, C. D., and Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. *Adv. Neural Inf. Process. Syst.*

[115] Sosa, D. N., Derry, A., Guo, M., Wei, E., Brinton, C., and Altman, R. B. (2020). A Literature-Based knowledge graph embedding method for identifying drug repurposing opportunities in rare diseases. *Pac. Symp. Biocomput.*, 25:463–474.

[116] Srinivasan, A. (2001). The aleph manual.

[117] Stanfield, Z., Coşkun, M., and Koyutürk, M. (2017). Drug response prediction as a link prediction problem. *Sci. Rep.*, 7(1):40321.

[118] Sturtevant, N. R. and Felner, A. (2018). A brief history and recent achievements in bidirectional search. In *Conference on Artificial Intelligence (AAAI)*, pages 8000–8007. webdocs.cs.ualberta.ca.

[119] Suh, B., Convertino, G., Chi, E. H., and Pirolli, P. (2009). The singularity is not near: slowing growth of wikipedia. In *Proceedings of the 5th International Symposium on Wikis and Open Collaboration - WikiSym '09*, New York, New York, USA. ACM Press.

[120] Sun, Z., Deng, Z.-h., Nie, J.-y., and Tang, J. (2019). Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations (ICLR)*, pages 1–18.

[121] Toutanova, K. and Chen, D. (2015). Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66.

[122] Trouillon, T., Dance, C. R., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. (2017). Knowledge graph completion via complex tensor factorization. *arXiv preprint arXiv:1702. 06879*.

[123] Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., and Bouchard, G. (2016). Complex Embeddings for Simple Link Prediction. In *International Conference on Machine Learning*, volume 48.

[124] Tzoumas, N., Farrah, T. E., Dhaun, N., and Webb, D. J. (2020). Established and emerging therapeutic uses of PDE type 5 inhibitors in cardiovascular disease. *Br. J. Pharmacol.*, 177(24):5467–5488.

[125] Vanhaelen, Q., Mamoshina, P., Aliper, A. M., Artemov, A., Lezhnina, K., Ozerov, I., Labat, I., and Zhavoronkov, A. (2017). Design of efficient computational workflows for in silico drug repurposing. *Drug Discov. Today*, 22(2):210–222.

[126] Vrandečić, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

[127] Wang, D., Cui, C., Ding, X., Xiong, Z., Zheng, M., Luo, X., Jiang, H., and Chen, K. (2019a). Improving the virtual screening ability of Target-Specific scoring functions using deep learning methods. *Front. Pharmacol.*, 10:924.

[128] Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. (2019b). Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.

[129] Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

[130] Wang, Q., Wang, B., and Guo, L. (2015a). Knowledge base completion using embeddings and rules. In *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*.

[131] Wang, X., Wang, D., Xu, C., He, X., Cao, Y., and Chua, T.-S. (2019c). Explainable reasoning over knowledge graphs for recommendation. *Conference on Artificial Intelligence (AAAI)*, 33:5329–5336.

[132] Wang, Y., Xing, J., Xu, Y., Zhou, N., Peng, J., Xiong, Z., Liu, X., Luo, X., Luo, C., Chen, K., Zheng, M., and Jiang, H. (2015b). In silico ADME/T modelling for rational drug design. *Quarterly reviews of biophysics*, 48(4):488–515.

[133] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2021). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24.

[134] Xiong, W., Hoang, T., and Wang, W. Y. (2017). DeepPath : A Reinforcement Learning Method for Knowledge Graph Reasoning. In *Annual Meeting of the Association for Computational Linguistics*, pages 564–573.

[135] Xiong, Z., Wang, D., Liu, X., Zhong, F., Wan, X., Li, X., Li, Z., Luo, X., Chen, K., Jiang, H., and Zheng, M. (2020). Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 63(16):8749–8760.

[136] Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations (ICLR)*, pages 1–13.

[137] Yang, F. and Cohen, W. W. (2017). Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In *Conference on Neural Information Processing Systems*.

[138] Yang, L. and Agarwal, P. (2011). Systematic drug repositioning based on clinical side-effects. *PLoS One*, 6(12):e28025.

[139] Ying, R., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. (2019). GNNExplainer: Generating explanations for graph neural networks. *Adv. Neural Inf. Process. Syst.*, 32:9240–9251.

[140] Zeng, Q., Patel, J., and Page, D. (2014). QuickFOIL: Scalable Inductive Logic Programming. *International Conference on Very Large Data Bases (VLDB)*, 8(3):197–208.

[141] Zhang, D., Cui, M., Yang, Y., Yang, P., Xie, C., Liu, D., Yu, B., and Chen, Z. (2019a). Knowledge graph-based image classification refinement. *IEEE Access*, 7:57678–57690.

[142] Zhang, P., Wang, F., Hu, J., Watson, I. B. M. T. J., and York, N. (2014). Towards Drug Repositioning : A Unified Computational Framework for Integrating Multiple Aspects of Drug Similarity and Disease Similarity. *AMIA Annual Symposium Proceedings*, 2014:1258–1267.

[143] Zhang, Q., Zhang, L., Qin, C., Wang, C., Zhu, H., Xiong, H., Chen, E., Guo, Q., and Zhuang, F. (2020). A survey on knowledge graph-based recommender systems. *Sci. Sin. Inf.*, 50(7):937–956.

[144] Zhang, S., Tay, Y., Yao, L., and Liu, Q. (2019b). Quaternion Knowledge Graph Embeddings. In *Conference on Neural Information Processing Systems (NeuralPS)*, pages 1073–1080.

[145] Zhao, S. and Li, S. (2012). A co-module approach for elucidating drug-disease associations and revealing their molecular basis. *Bioinformatics*, 28(7):955–961.

[146] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., and Sun, M. (2019). Graph Neural Networks: A Review of Methods and Applications. In *Arxiv*, pages 1–20.

[147] Zhou, Y., Hou, Y., Shen, J., Huang, Y., Martin, W., and Cheng, F. (2020a). Network-based drug repurposing for novel coronavirus 2019-nCoV/SARS-CoV-2. *Cell Discov*, 6:14.

[148] Zhou, Y., Hou, Y., Shen, J., Mehra, R., Kallianpur, A., Culver, D. A., Gack, M. U., Farha, S., Zein, J., Comhair, S., Fiocchi, C., Stappenbeck, T., Chan, T., Eng, C., Jung, J. U., Jehi, L., Erzurum, S., and Cheng, F. (2020b). A network medicine approach to investigation and population-based validation of disease manifestations and drug repurposing for COVID-19. *PLoS Biol.*, 18(11):e3000970.

[149] Zhou, Y., Wang, F., Tang, J., Nussinov, R., and Cheng, F. (2020c). Artificial intelligence in COVID-19 drug repurposing. *The Lancet*, 7500(20).

[150] Zhu, Z., Xu, S., Qu, M., and Tang, J. (2019). Graphvite: A high-performance cpu-gpu hybrid system for node embedding. In *The World Wide Web Conference*, pages 2494–2504. ACM.

[151] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2021). A comprehensive survey on transfer learning. *Proc. IEEE*, 109(1):43–76.

[152] Zitnik, M., Agrawal, M., and Leskovec, J. (2018). Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466.

[153] Zitnik, M., Nguyen, F., Wang, B., Leskovec, J., Goldenberg, A., and Hoffman, M. M. (2019). Machine learning for integrating data in biology and medicine : Principles , practice , and opportunities. *Information Fusion*, 50(September 2018):71–91.

[154] Zou, X. (2020). A survey on application of knowledge graph. *J. Phys. Conf. Ser.*, 1487(1):012016.