A Survey on Rule-Based Knowledge Graph Reasoning

Authors: Zeng Zefan, Cheng Qing, Si Yuehang, Liu Zhong, Cheng Qing

Date: 2023-05-22T00:00:00+00:00

Abstract

Objective: Currently, rule-based knowledge graph reasoning methods are continuously emerging, yet systematic categorization and analysis are lacking. This paper systematically reviews research work related to rule-based knowledge graph reasoning.

Methods: We elaborate on and analyze the core ideas, key techniques, advantages, and disadvantages of relevant classical methods, and compare the performance of different types of approaches.

Results: This paper identifies four categories of methods: those based on inductive logic programming, those based on probabilistic graphical models and rules, those based on embedding representations and rules, and those based on neural networks and rules.

Limitations: Specific applied research on these methods is currently relatively limited.

Conclusion: Rule-based knowledge graph reasoning methods perform inference by mining underlying logical rules, demonstrating strong generalization capabilities and interpretability. Hybrid rule-based reasoning methods still hold significant promise for future applications.

Full Text

A Survey on Rule-Based Knowledge Graph Reasoning

ZENG Ze-fan[1],[2], CHENG Qing[1], SI Yue-hang[1], LIU Zhong[1] [1](Big Data and Decision Laboratory, National University of Defense Technology, Changsha 410073) [2](Fujian Province Military Region, Fuzhou 350001)

Abstract: [Objective] Currently, rule-based knowledge graph reasoning methods are constantly emerging, yet systematic classification and analysis remain

lacking. This article surveys research work related to rule-based knowledge graph reasoning. [Methods] We elaborate and analyze the core ideas, key technologies, advantages, and disadvantages of relevant classical methods, and compare the performance of different types of approaches. [Results] This paper summarizes four categories of methods, including inductive logic programming-based methods, probabilistic graphical model and rule-based methods, embedding representation and rule-based methods, and neural network and rule-based methods. [Limitations] Research on the specific applications of relevant methods is still relatively scarce. [Conclusions] Rule-based knowledge graph reasoning methods perform inference by mining underlying logical rules, offering strong generalization capability and interpretability. Hybrid rule-based reasoning methods still hold considerable promise for future applications.

Keywords: knowledge graph reasoning, rules, inductive logic programming, probability graph, embedding representation, neural network

Classification: TP391

Knowledge representation and learning are essential means for humans to understand the world, acquire skills, and make decisions. The era of big data has given rise to massive datasets rich in knowledge, creating a significant challenge: how to leverage this knowledge to assist humans in problem-solving. In 2012, Google first introduced the concept of the Knowledge Graph (KG), a simple yet effective knowledge representation method containing abundant human knowledge resources [1]. In recent years, rapid development in Internet technology and Web applications has led to vast amounts of data being published online, serving as an important source for large-scale knowledge extraction and knowledge graph construction. With the explosive growth of open knowledge bases such as WordNet [2], DBpedia [3], YAGO [4], NELL [5], Probase [6], CN-DBpedia [7], and zhishi.me [8] both domestically and internationally, knowledge graphs have been widely used to support practical applications of semantic networks, including recommendation systems [9], question-answering systems [10], and intelligent dialogue [11].

Based on differences in entity types and elements, knowledge graphs can be categorized into three types: Static Knowledge Graphs (SKG), Temporal Knowledge Graphs (TKG), and Multi-Modal Knowledge Graphs (MMKG) [12]. The most common and intensively researched among these is the static knowledge graph. In most research, when static knowledge graphs are mentioned without reference to the other two types, they are simply referred to as knowledge graphs. The research object of this paper is static knowledge graphs; for convenience, all subsequent references to "knowledge graph" in this paper denote "static knowledge graph."

Due to the incompleteness of knowledge sources and biases in the knowledge extraction process, most current knowledge graphs suffer from missing or erroneous information to varying degrees. Consequently, Knowledge Graph Reasoning (KGR) technology [13] has emerged. Knowledge graph reasoning is an

inference method that derives new facts from existing ones within a knowledge graph, widely applied in artificial intelligence tasks such as intelligent question answering, recommendation systems, dialogue generation, information extraction, and image classification [14-18], playing a crucial role in fields like military, finance, transportation, and cybersecurity [19-22].

To achieve the leap from "computational intelligence" and "perceptual intelligence" to "cognitive intelligence" [23], eXplainable Artificial Intelligence (XAI) has received increasing attention and emphasis in the academic research community, 被视为人工智能继续稳步发展的必要条件. Although currently popular distributed representation-based knowledge graph reasoning methods [24,25] offer high efficiency, their operations in latent spaces and "black-box" model characteristics pose significant challenges to intuitively understanding reasoning bases and decision-making processes. Consequently, such methods are difficult to apply in domains with high reliability requirements (such as military, medical, and information security).

Rule-based reasoning methods directly derive new facts by mining underlying logical rules, offering not only guaranteed reasoning accuracy but also strong interpretability that enables users to more intuitively understand the reasoning process, providing reliable guidance for decision support. Additionally, rule-based knowledge reasoning can perform inference on unseen facts (i.e., inductive reasoning). Early rule-based knowledge graph reasoning relied on prior knowledge, hard matching, and exhaustive search, resulting in limited rule expressiveness, poor scalability and robustness, and low efficiency [26]. With the maturation of deep learning and reinforcement learning techniques [27,28], an increasing number of studies combine these technologies for rule learning or utilize rules to assist relevant reasoning methods, aiming to achieve efficient, accurate, transferable, and interpretable knowledge graph reasoning.

Large Language Models (LLM), led by ChatGPT, provide new tools for knowledge reasoning [29-31]. LLMs can learn vast amounts of knowledge and achieve efficient knowledge graph reasoning by leveraging rich factual knowledge from pre-trained corpora or external data retrieval. However, LLMs also face potential risks from "hallucination" [32]—i.e., incorrect or confused information. Learning trustworthy rules and incorporating them as "instructions" into large models to develop neural-rule hybrid reasoning models that assist LLMs in completing knowledge reasoning holds promise for further enhancing the reliability, understandability, and domain-specific adaptability of large models.

Currently, scholars have published multiple survey articles on knowledge graph reasoning. Among them, reference [12] comprehensively reviews research on static, temporal, and multi-modal knowledge graph reasoning; reference [33] surveys static knowledge graph reasoning by distinguishing between multi-shot, few-shot, and single (zero)-shot scenarios; reference [34] systematically introduces neural network-based knowledge graph reasoning methods; reference [35] collects and summarizes methods and key technologies for applying graph neural networks to knowledge graph reasoning; reference [36] focuses on representation

learning-based knowledge graph reasoning, summarizing and comparing relevant methods; references [23] and [26] examine different knowledge graph reasoning techniques and their explanation methods from an interpretability perspective; reference [37] surveys research progress on deep reinforcement learning-based knowledge graph reasoning; reference [38] categorizes hybrid symbolic and statistical knowledge graph reasoning methods; reference [39] reviews how to combine logical rules with embedding techniques for knowledge graph reasoning; and reference [40] surveys symbolic reasoning, neural reasoning, and neuro-symbolic hybrid reasoning methods from the perspectives of symbolic and connectionist paradigms.

This paper focuses on rule-based knowledge graph reasoning methods as its primary research object, organizing and reviewing literature related to rule-based knowledge graph reasoning. After introducing background knowledge and basic concepts, we elaborate in detail on knowledge graph reasoning methods based on inductive logic programming, probabilistic graphical models and rules, embedding representation and rules, and neural networks and rules. The classification framework is shown in [Figure 1: see original paper]. We emphasize the core ideas, key technologies, and reasoning effects of typical works in each category, conduct comparative analyses of different types of methods, and summarize main challenges and future research directions, aiming to provide reference for subsequent research.

2 Background Knowledge and Concept Definitions

This section introduces background knowledge and basic concepts related to knowledge graphs, knowledge graph reasoning, and rule-based knowledge reasoning.

2.1 Knowledge Graph

A knowledge graph is a graphical knowledge representation method—a giant semantic network composed of nodes and edges, where nodes represent concepts or entities in the physical world, and edges represent topological connections and semantic relationships between nodes.

Reference [12] categorizes knowledge graphs into static knowledge graphs, temporal knowledge graphs, and multi-modal knowledge graphs. Temporal knowledge graphs contain timestamp information, while multi-modal knowledge graphs contain multi-modal data such as images and text in their entity types.

Knowledge graphs typically represent facts as triples (head entity, relation, tail entity), with vocabulary defined in a schema (also called an ontology) that indicates how two entities are connected by a specific relation. For example, (Nunez, PlayforTeam, Liverpool) indicates that the athlete Nunez plays for the Liverpool football team.

In symbolic representation, a knowledge graph is defined as a set of entities,

relations, and facts. A triple (e_i, r, e_j) represents a fact F, where $e_i, e_j \in \mathcal{E}$, $r \in \mathcal{R}$, and $F \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Typically, $(e_i, r, e_j) \in \mathcal{F}$. During knowledge graph construction, if $(e_i, r, e_j) \in \mathcal{F}$, then (e_j, r^{-1}, e_i) is also added to $\mathcal{F}$. The superscript "-1" denotes the inverse relation. [Figure 2: see original paper] shows an example of a knowledge graph.

In graph model representation, a knowledge graph can be viewed as a directed labeled multigraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \mathcal{E}$ and $\mathcal{E} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. [Figure 2: see original paper] presents a simple knowledge graph case.

2.2 Knowledge Graph Reasoning

Knowledge graph reasoning derives hidden or new facts from existing knowledge, essentially a process of generalizing individual knowledge to general knowledge through deduction or induction [41]. Based on whether object triples are complete, knowledge graph reasoning tasks can be divided into knowledge graph denoising and knowledge graph completion. The former judges the correctness of existing complete triples, while the latter predicts missing entities or relations.

In symbolic representation, given a knowledge graph $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, knowledge graph reasoning uses known facts to infer query facts (e_h, r, e_t). Knowledge graph denoising determines whether (e_h, r, e_t) is correct, while knowledge graph completion includes three subtasks: head entity completion $(?, r, e_t)$, tail entity completion $(e_h, r, ?)$, and relation prediction $(e_h, ?, e_t)$. [Figure 3: see original paper] provides an example of knowledge graph reasoning, where triples related to red arrows correspond to denoising tasks, those related to black dashed arrows correspond to relation prediction tasks, and blue arrows with their pointed entities correspond to tail entities that need completion.

2.3 Rule-Based Knowledge Graph Reasoning

Rule-based knowledge reasoning methods mine reliable rules for knowledge graph reasoning. The basic unit of a rule is called an atom, typically denoted by α. An atom can be defined as a unary or multi-ary (usually binary) predicate symbol that maps a series of variables to True or False. Specifically, if all variables in an atom are constants, the atom is called a grounding atom. Multiple atoms connected by logical connectives constitute a rule. Rules typically take the form $body \Rightarrow head$, where the rule body consists of the conjunction of multiple atoms representing the preconditions for rule validity, and the rule head is a single atom containing the target predicate representing the conclusion drawn from the rule.

First Order Logic (FOL) [59] is a primary form of knowledge representation and reasoning, transforming triples in knowledge graphs into "binary predicate + constant" combinations, i.e., $< subject, predicate, object > \equiv predicate < subject, object >$. For example, in [Figure 3: see original paper], knowing the rule $BornInCity(x, y) \wedge Nationality(y, z) \Rightarrow Nationality(x, z)$, one can infer that Malala Yousafzai's nationality is Pakistan.

Horn Rule Logic [58], also called Horn clauses or Horn rules, is a subset of first-order logic characterized by simple form and ease of description, making it a widely applicable rule representation method. Horn rules consist of a head and a body, where the head is a single atom and the body is the conjunction of several atoms. A Horn rule H can be expressed as: $\alpha_1 \wedge \alpha_2 \wedge ... \wedge \alpha_n \Rightarrow \alpha$, where α is called the head atom and $\alpha_1, ..., \alpha_n$ are body atoms. A represents the rule body, connecting body atoms using conjunction or disjunction paradigms. The rule body is also called a formula, meaning that if A is true, then the rule head atom on the right is also true. Closed rules in Horn logic refer to rules where all variables appear in at least two atoms, which is also the most common type of rule in knowledge graph reasoning.

Rule-based knowledge reasoning is a generalization of the rule paradigm, semantically consistent with natural language expression and human thinking patterns. It can provide clear and explicit reasoning bases, offering strong interpretability and high reasoning precision, making it a globally interpretable knowledge graph reasoning method. Such methods were also the earliest used for knowledge graph reasoning. However, due to the semantic diversity of knowledge graphs, rules involved in reasoning are often complex, leading to problems in traditional rule learning and knowledge reasoning methods such as poor scalability, limited expressiveness, and high time and space complexity [26]. To address these issues, researchers have attempted to combine graph models and neural networks with rules, leveraging their advantages in transferability, robustness, and efficiency for hybrid reasoning. In recent years, hybrid reasoning has become one of the research hotspots in knowledge graph reasoning.

3 Knowledge Graph Reasoning Based on Inductive Logic Programming

Inductive Logic Programming (ILP) is a rule learning method based on first-order logic [40]. This approach aims to find underlying patterns composed of logical programs, rules, or formulas shared in data. ILP constrains logical rules to Horn rules, then constructs rules that can explain positive examples while rejecting negative examples based on given predicates, ground atoms, positive instances, and negative instances. ILP-based knowledge graph reasoning methods can be divided into first-order inductive learning-based methods, association rule mining-based methods, and relation path sampling-based methods.

3.1 First-Order Inductive Learning Methods

First Order Inductive Learner (FOIL) is a domain-independent rule learning method [42] that aims to learn reasoning rules from text. Based on the SHERLOCK system [43], the FOIL method learns rules through the following four steps: (1) Identify valid classes and their instances; (2) Discover relationships between classes; (3) Use relationships between classes to learn reasoning rules and determine their confidence scores; (4) Use the HOLMES reasoning engine

to find facts through logical reasoning and estimate the confidence of each fact using Markov networks. The specific workflow of FOIL is shown in [Figure 4: see original paper].

Landwehr et al. [44] integrated naive Bayes learning methods into FOIL, proposing two dynamic propositionalization rule learning methods—Naive Bayesian First Order Inductive Learner (N-FOIL) and Tree-augmented Naive Bayesian First Order Inductive Learner (T-FOIL). N-FOIL does not use a divide-and-conquer approach, maintaining the same computational complexity as FOIL. However, after each iteration, the number of instances decreases, and paths are adjusted based on scoring of additional clauses (rules), effectively increasing the diversity of rule learning. To prevent excessive model parameters, N-FOIL makes the strong assumption that "the probabilities of instances satisfying different queries are independent of each other," i.e.: $\lambda P(p|i_1, ..., i_n) = \prod_{q=1}^{n} P(i_q|p)$, where P represents query probability, p is the class label, and i_q is a query. However, this assumption is often difficult to satisfy in practice. T-FOIL addresses this issue by introducing additional dependency trees to reduce the number of parameters, relaxing Equation 2 to: $\lambda P(p|i_1, ..., i_n) = P(p) \prod_{q=1}^{n} P(i_q|pa_q)$, where pa_q represents the additional parent node of node i_q in the tree-augmented naive Bayes model. The tree-augmentation strategy reduces model parameters from $O(2^{\#\text{nodes}})$ to $O(\#\text{nodes})$, effectively decreasing computational complexity and improving learning efficiency.

Landwehr et al. [45], inspired by the relationship between kernel learning and function learning, drew an analogy between kernel function learning and the relationship between structure and parameter learning in ILP statistical relational learning. They then combined kernel learning techniques with the FOIL algorithm to propose the K-FOIL (Kernel First Order Inductive Learner) method. In K-FOIL, structure learning corresponds to inferring an appropriate kernel on logical objects, while parameter learning corresponds to function learning in reproducing kernel Hilbert spaces. Subsequently, a discrete space search algorithm is used to induce kernel function K in a logical setting, and K is utilized to perform multi-task statistical logical rule learning. Case analysis shows that K-FOIL offers advantages over both FOIL and N-FOIL in terms of accuracy and efficiency.

Additionally, Zeng et al. [46] proposed the QuickFOIL algorithm. Based on top-down greedy ILP search, this algorithm employs a scoring function that drives the heuristic search process and a new pruning strategy, enabling the algorithm to find high-quality rules. When constructing clauses, it starts from an empty text and then uses a greedy heuristic method to add one new literal at a time, followed by pruning key decisions in the partial search space. QuickFOIL reduces the number of join operations from $O(n^2)$ to $O(n)$. Furthermore, QuickFOIL develops various query optimization and caching techniques for multiple class queries to accelerate query speed.

FOIL-based methods can achieve accurate and efficient reasoning in small-scale

knowledge graphs. However, in medium-to-large-scale knowledge graphs, entity and relation types become more diverse, and the number of rules increases dramatically, causing computational complexity to rise exponentially and significantly reducing accuracy and efficiency. Moreover, FOIL methods typically require negative instances, while the Open World Assumption (OWA) of knowledge graphs means that missing data cannot be used as negative instances.

3.2 Association Rule Mining Methods

To address the problems of FOIL methods and improve the induction efficiency of reasoning methods, Galárraga et al. [47] proposed AMIE (Association Mining under Incomplete Evidence), a system for mining Horn rules in large RDF (Resource Description Framework) knowledge bases. AMIE can mine Horn rules in large RDF knowledge bases. It generates a rule base based on relation types in the knowledge graph, then finds instances satisfying these rules from the knowledge graph, and calculates rule confidence. If the confidence exceeds a certain threshold, the rule is considered reliable.

AMIE's rule learning includes two stages. The first stage is rule discovery, which treats rules as sequences of atoms. It initializes an empty rule set and uses three mining operators to add rules: (1) Add Dangling Atom (DO), i.e., add new variables to existing rules; (2) Add Instantiated Atom (IO), i.e., add constant atoms to existing rules; (3) Add Closing Atom (CO), i.e., add a new atom to prevent further rule expansion. The second stage is rule pruning, which calculates confidence based on the Partial Completeness Assumption (PCA). If adding an atom to a rule does not improve confidence, the rule is no longer expanded. If a rule only has a head relation, it checks whether its confidence exceeds threshold θ; if not, it is directly deleted. AMIE does not require knowledge beyond the knowledge graph, nor does it need configuration or parameter tuning, showing significant improvements over FOIL series methods in runtime, number of output rules, and rule quality.

The AMIE system reduces search space by constraining rule length, type, and confidence metrics. However, since all relations can be connected to any other relation through rules during rule discovery, it still faces the problem of search space explosion when dealing with large-scale knowledge graphs. To address these limitations, Galárraga et al. [48] proposed an improved version of AMIE called AMIE+. AMIE+ optimizes in three main aspects: (1) In the rule discovery step, AMIE+ only adds CO and IO in the final step, not DO, to avoid unclosed rules. (2) In the rule pruning step, it proposes an approximate confidence calculation method for computing confidence of rules with high complexity. (3) It combines discovered rules with heuristic type checking and joint reasoning: type checking adds head and tail entity category information in the rule body to improve reasoning accuracy; joint reasoning means that for a given query, if multiple rules can infer candidate answers, it aggregates the PCA confidence of these rules to improve the ranking of reasoning results.

Although AMIE+ shows significant improvements in accuracy and efficiency over AMIE, its algorithm can only mine one rule at a time during execution, making parallelization impossible. To address this issue, Lajus et al. [49] further improved it by proposing the AMIE3 system. This system introduces multiple complex pruning strategies, parallel optimization algorithms, and lazy confidence calculation to accelerate strategy mining, making it more suitable for large-scale knowledge graphs. Additionally, AMIE3 can exhaustively mine all rules above support and confidence thresholds without resorting to sampling or approximation.

Furthermore, Wang et al. [50] proposed a rule learning method called RDF2Rules, which generates association rules by mining Frequent Predicate Cycles (FPC). This method defines predicate cycles (where an entity returns to the original entity through multiple predicates) as $x_0 \xrightarrow{p_1^{d_1}} x_1 \xrightarrow{p_2^{d_2}} ... \xrightarrow{p_n^{d_n}} x_n$, where x_i represents entities, p_i represents predicates, and $d_i \in \{-1, 1\}$ indicates predicate direction. It determines whether a cycle is an FPC through support. Then, it uses FPC to generate and evaluate rules, employing entity type information during the process to make learned rules more accurate.

However, association rule mining methods often require traversing all relation paths, resulting in high computational complexity and low efficiency when applied to large-scale knowledge graphs.

3.3 Relation Path Sampling Methods

To reduce the computational complexity of rule search, scholars have proposed relation path sampling-based methods. The basic idea of this approach is that for each target relation, only triples related to that relation in the knowledge graph are sampled, and then the sampling information is used for rule learning. Representative methods of this type include RuleN [51], AnyBURL [52] and its variants, and C-NN [55].

RuleN is a rule mining method based on double random sampling, supporting two types of rule mining: nP-type rules and C-type rules, with forms: $r(x, y) \leftarrow r_1(x, a_1) \wedge ... \wedge r_n(a_{n-1}, y)$ and $r(x, y) \leftarrow r_1(x, y) \wedge ... \wedge r_n(x, y)$. For nP-type query relation r, this method only extracts K triples (x, r, y) related to it, then uses a depth-first traversal algorithm to determine all paths of length n between x and y as candidate rule bodies. It randomly samples paths satisfying nP-type candidate rule bodies and checks whether relation r exists between head and tail entities in the path. For C-type rules, it similarly randomly selects K triples (x, r, y), then creates rule $\forall x, y : r(x, y) \leftarrow r_1(x, y) \wedge ... \wedge r_n(x, y)$ for each triple. Finally, it calculates approximate confidence of path rules by randomly sampling triples whose rule bodies are true. For C-type rules, it only needs to select a sample from r facts and calculate the frequency of finding x or y in subject and object positions as confidence. The RuleN method effectively reduces search space and simplifies computation through sampling.

Ferré proposed the Concepts of Nearest Neighbours (C-NN) method. C-NN is a rule learning method based on entity clustering that does not require large-scale training or retraining, avoiding high costs in runtime and memory. C-NN introduces a symbolic form of K-nearest neighbors, replacing traditional KNN's numerical distance with graph patterns that provide an understandable representation of the similarity degree between two entities. The screening of nearest neighbors for each entity is essentially also a sampling process. In rule learning, clusters of similar entities are identified based on common graph patterns, and different rules are derived from these graph patterns and combined for reasoning. C-NN can provide explanations for each inference.

Meilicke et al. proposed AnyBURL (Anytime Bottom-Up Rule Learning), a bottom-up technique capable of learning logical rules from large-scale knowledge graphs at any time. The basic idea of the method is to sample paths of length n from a given knowledge graph G, starting from $2n = \text{max\_path\_len}$, and learn rules of length $n-1$ from paths of length n, where the first edge on the path corresponds to the head atom. After reaching a certain saturation SAT for rules of length $n-1$, it sets $n = n+1$ to learn longer rules. Here, saturation is defined as $SAT = \frac{|SR'|}{|SR|}$, where SR represents all rules discovered in the current time span, and SR' represents rules discovered in the current time span that coincide with rules discovered in past iterations. AnyBURL's rule learning process occurs in a sequence of time spans of length st. Within a time span, the algorithm learns as many rules as possible by iteratively sampling random paths. After the time span ends, confidence scores are assigned to rules. Finally, candidate rules are ranked by generating the maximum confidence among all candidate rules. This method has high computational efficiency and low computational resource requirements, can generate explanations based on proposed candidate rules, and significantly outperforms other rule-based reasoning methods, showing great potential in performance and interpretability.

AnyBURL's mined rules depend on the setting of saturation thresholds, but learning reasonable thresholds is often difficult, leading to poor robustness of the method. To solve this problem, Meilicke et al. [53] proposed the Reinforce-AnyBURL algorithm. This algorithm uses reinforcement learning methods, treating rule interpretability, confidence, and length as rewards for rule learning, and proposes an object identification method that adds constraints to rules to remove redundant and duplicate rules.

To prevent rule aggregation from being affected by redundant information, Ott et al. [54] further improved the AnyBURL method by proposing a transferable fast non-redundant rule mining method called SAFRAN (Scalable And Fast non-redundant Rule Application). This method uses a non-redundant noise aggregation approach that detects and clusters redundant rules before aggregation, thereby mitigating their negative impact and improving prediction performance. SAFRAN uses a maximum aggregation method to predict entity confidence scores, then further aggregates predictions from different clusters based on noise or methods. Additionally, SAFRAN employs both grid search

and random search methods to determine the optimal clustering threshold for triple rules based on type combinations. Compared with AnyBURL, SAFRAN mines higher-quality rules and produces more accurate reasoning results.

provides information on reasoning methods based on inductive logic programming. ILP-based rule learning and knowledge graph reasoning methods are "hard" reasoning methods, where mined rules are judged correct by comparing confidence scores with thresholds. However, such methods are somewhat subjective due to threshold settings and only apply to precise reasoning, unable to represent uncertain information.

4 Knowledge Graph Reasoning Based on Probabilistic Graphical Models and Rules

To achieve fuzzy reasoning, researchers have combined probabilistic graphical models with rule learning, proposing various "soft" reasoning methods suitable for fuzzy inference, mainly including methods based on Markov Logic Networks, inference trees, and probabilistic soft logic.

4.1 Markov Logic Network-Based Methods

The Markov Logic Network (MLN) proposed by Richardson et al. [56] establishes a probabilistic graphical model based on predefined rules and knowledge graph facts, then learns weights for different rules. Given any rule set $\{\gamma_i\}$, each rule in the set corresponds to instances in the knowledge graph. Based on $\{\gamma_i\}$, a Markov logic network can be established.

The main steps of MLN modeling are: (1) Build a node for each ground atom in each rule, setting the node value to 1 if the atom is observed in the knowledge graph, otherwise 0; (2) Establish edges between two nodes when two ground atoms can simultaneously instantiate at least one rule; (3) All ground atoms form a clique corresponding to a feature, with value 1 if the basic rule is true, otherwise 0.

$P(X = x) = \frac{1}{Z} \exp(\sum_i w_i n_i(x))$

where $n_i(x)$ is the number of true ground atoms of rule γ_i, and w_i is the weight corresponding to rule γ_i. Then, Markov Chain Monte Carlo (MCMC) algorithms are applied for inference in MLN, and weights w_i are effectively learned by optimizing pseudo-likelihood measures. Based on w_i, given adjacent ground atoms, one can infer the probability of $Lives\_in(LeBron, L.A)$.

[Figure 5: see original paper] shows an example of MLN corresponding to two rules. MLN was the first model developed for fuzzy reasoning on knowledge graphs. However, due to the complex graph structure between triples, MLN's inference process is difficult and inefficient. Moreover, missing triples in knowledge graphs also affect reasoning results through rules.

4.2 Inference Tree-Based Methods

Selective Linear Definite (SLD) [57] is a method for knowledge graph reasoning that constructs inference trees for queries. Compared with MLN, SLD inference trees can effectively alleviate the problem of complex triple structures and offer higher reasoning efficiency. The standard SLD method builds inference trees in a top-down manner, as shown in [Figure 6: see original paper]. It first uses the query to initialize the root node, then recursively creates sub-goals by applying each clause and its instantiation.

Based on SLD inference trees, Raedt et al. [57] proposed the ProbLog method, a probabilistic extension of the logic programming method Prolog. Compared with Prolog, ProbLog adds a probability p_i to each clause c_i, where c_i represents a rule or a ground atom used to derive query examples, and finally obtains reasoning results through recursion based on inference trees.

$Lives\_in(Y,Z) \rightarrow Lives\_in(X,Y); 0.8 : Part\_of(X,Y) \wedge Located\_in(Y,Z)$ $0.7 : Spouse\_of(X,Y) \wedge Lives\_in(Y,Z) \rightarrow Lives\_in(X,Y); 1.0 : Spouse\_of(LeBron, Savannah)$. Given any query q, $P(q|T)$ is decomposed into calculating the sum of joint probabilities of all possible clause sets, i.e.:

$$P_s(q|T) = \sum_{L \subseteq T} P(L)P(q|L)$$

where $P(L)$ describes the probability distribution of causes L, and $P(q|L)$ represents the probability of query q given cause set L, with value 1 if at least one answer can instantiate L and make the query true, otherwise 0. To calculate $P(q|L)$, ProbLog uses SLD to construct an inference tree for q, first initializing the root node according to the query, then recursively creating sub-goals by applying each clause and its instantiation, stopping when reaching termination conditions (finding an answer or reaching maximum tree depth).

To improve reasoning efficiency, Cussens [60] proposed the Stochastic Logic Programs (SLP) method. SLP is a logic program with parameterized clauses that defines a log-linear distribution for goal inversion, which provides distributions bound to variables through marginalization, enabling SLP to represent complex rule distributions. SLP defines a stochastic process traversing SLD trees, where the probability distribution defined on nodes is learned by up-weighting required answer clauses and down-weighting other clauses. Additionally, Cussens proposed a Fault-Adjusted Maximisation (FAM) method, an instance of the EM algorithm specifically applied to normalized SLPs, providing a closed form for computing parameter updates in iterative maximization methods, effectively improving rule optimization speed.

Wang et al. [61] utilized the biased sampling strategy of the PPR (Personalized PageRank) algorithm to replace SLP's random sampling strategy, proposing the ProPPR method. Unlike directly setting a probability for $P(L|T)$ in Equation 6, ProPPR uses predefined features in PPR to calculate the probability of each clause, then designs a local grounding process based on local partitioning

methods to achieve inference. This local grounding process has query capabilities that can divide rule weight learning tasks into multiple separate gradient calculations, enabling parallel learning to improve rule learning and reasoning efficiency. Moreover, experiments show that ProPPR performs well on entity resolution tasks even without weight learning, while supervised weight learning improves accuracy.

In addition to ProbLog, SLP, and PPR, methods such as Datalog [62] and MV-Datalog±View [63] also adopt reasoning logic similar to SLD. Although inference tree-based methods improve speed compared to MLN, they still have unsatisfactory reasoning accuracy and speed when facing knowledge graphs with large numbers of missing triples.

4.3 Probabilistic Soft Logic-Based Methods

To address knowledge graph reasoning problems with missing triples, Kimmig et al. [64] proposed Probabilistic Soft Logic (PSL). PSL is a framework for collective probabilistic reasoning in relational domains, consisting of a set of first-order logic rules, bodies, and single literal heads.

Similar to ProbLog, PSL rules are assigned weights. To determine the degree to which basic rules are satisfied, PSL uses the Lukasiewicz t-norm [65] and its corresponding co-norm as relaxations for logical AND and OR:

$I(\neg a) = 1 - I(a)\ I(a \wedge b) = \max(0, I(a) + I(b) - 1)\ I(a \vee b) = \min(1, I(a) + I(b))$
$I(a \rightarrow b) = \min(1, 1 - I(a) + I(b))$

"$\sim$" represents relaxation from Boolean domain, I represents a given interpretation, and λ represents atoms. Through the above formulas, Boolean truth values are relaxed to continuous soft truth values in the interval [0,1]. For a basic rule r, the interpretation I of atoms in r determines whether r is satisfied, and the "gap" between rule distance and satisfaction is calculated using $\mathrm{dist}_r(I) = \max(0, I(body) - I(head))$. Through relaxation and distance calculation, PSL integrates similarity functions into the graph model. Additionally, PSL restricts the syntax of first-order formulas to syntax with body rules. These two characteristics transform reasoning into a convex optimization problem in continuous space, enabling rule learning under missing knowledge and making solving more convenient.

To reduce knowledge graph identification and reasoning time and achieve knowledge graph reasoning under information system noise, Pujara et al. [66] proposed an ontology-aware partitioning method called OAP (Ontology-Aware Partitioning). OAP builds an ontology knowledge graph with entities and relations as nodes and ontology constraint relations as edges, as shown in [Figure 7: see original paper], and uses PSL to define joint probability distributions on the knowledge graph. Then, using minimal edge segmentation techniques, it partitions relations and labels existing in the ontology, creating corresponding partitions for relation and label-specific instance extraction to perform knowledge

reasoning. To address the imbalance in partitioning caused by the contribution of ontological information and inconsistencies between different relations and labels in data, OAP weights each vertex based on the extraction frequency of relations or labels, constraining graph cuts to produce clusters with equal vertex weights. OAP leverages ontology for partition extraction, preserving the advantages of joint methods for building knowledge graphs and significantly improving operational efficiency, enabling joint reasoning of knowledge graphs from noisy outputs of information extraction systems.

To adapt to large-scale complex knowledge graphs, Bach et al. [67] proposed the Hinge-Loss Markov Random Field (HL-MRF) model. HL-MRF is a probabilistic graphical model combining stochastic algorithms, probabilistic graphical models, and fuzzy logic communities. First, researchers proved that stochastic algorithms for MAX SAT, local consistency relaxation of discrete MRFs defined using Boolean logic, and continuous information inference with fuzzy logic can be unified under the same convex optimization objective. Then, the inference objective is generalized to a weighted sum of hinge-loss features. HL-MRF defines PSL through the connection between hinge-loss potentials and logical rules, making the model easily scalable to large knowledge graphs. Additionally, the syntax provided by PSL enables the model to be better understood and extended by users.

provides information on reasoning methods based on probabilistic graphical models and rules. Although these methods can achieve fuzzy reasoning, they essentially still rely on rule search and mining, requiring complex traversal calculations on knowledge graphs. Therefore, their accuracy, efficiency, and robustness in large-scale graphs are not strong, and they have not been widely applied in large-scale knowledge graphs.

5 Knowledge Graph Reasoning Based on Embedding Representation and Rules

Traditional explicit rule learning-based knowledge graph reasoning methods have good readability and interpretability, but these limited and discrete rules often struggle to completely describe all intrinsic relationships between data and are not robust to fuzziness and noise. Knowledge graph embedding representation maps entities and relations into dense vectors in low-dimensional continuous vector space, which can effectively represent and measure semantic associations between entities and relations, improve computational efficiency, alleviate data sparsity, and be more robust to fuzziness and noise. How to effectively combine the advantages of embedding representation and rule learning to improve reasoning capability has become one of the key research focuses. In existing research work, methods combining embedding representation and rules for knowledge graph reasoning can be divided into rule embedding representation learning-based reasoning methods and mutual enhancement-based reasoning methods between embedding representation and rules.

5.1 Rule Embedding Representation Learning Methods

These methods perform reasoning through rule embedding or rule-enhanced embedding representation. Rules can provide information for knowledge graph embedding representation, and triples inferred through rules can provide more training sets for embedding representation, improving the efficiency and quality of embedding representation. They can also provide more explanatory basis for reasoning results, enhancing reasoning interpretability.

Guo et al. [68] proposed the KALE method in 2016, one of the earliest studies combining embedding representation and rules for knowledge graph reasoning. KALE embeds both rules and triples into a unified framework for joint embedding modeling of triples and rules. The method treats triples as basic atoms. Given a logical rule, it first instantiates the rule with centralized entities, treating the rule as a complex formula composed of basic atoms and logical connectives. For triple modeling, it adopts the TransE [69] method, defining its truth value as: $f_r(h,t) = ||\mathbf{e}_h + \mathbf{r} - \mathbf{e}_t||$, where $\mathbf{e}_h, \mathbf{r}, \mathbf{e}_t \in \mathbb{R}^d$ are embedding vectors of entities and relations, d is the embedding dimension. For rule modeling, it adopts fuzzy logic methods, considering two types of rules: one is $r(x,y) \leftarrow r_1(x,z) \wedge r_2(z,y)$, denoted as $r \leftarrow r_1 \wedge r_2$; the other is $r(x,y) \leftarrow r_1(y,x)$, denoted as $r \leftarrow r_1^{-1}$. The truth value I is calculated as: $I(r \leftarrow r_1 \wedge r_2) = \min(f_{r_1}(x,z), f_{r_2}(z,y))$ and $I(r \leftarrow r_1^{-1}) = f_{r_1}(y,x)$. The larger the I value, the greater the probability that the rule is satisfied. Finally, it minimizes a global loss over the set of trainable formulas to learn embedding representations compatible with both triples and rules.

Wang et al. [70] drew inspiration from the ProPPR method and proposed a first-order logic embedding method based on matrix factorization called ProPPR-MF (ProPPR Matrix Factorization). This method first proposes a structure learning strategy based on structural gradients to generate reasonable inference formulas from facts. Then, it uses background facts, training examples, and inference formulas to build grounded proof graphs. Finally, it maps training examples to rows of a binary matrix and inference formulas to columns, using a transferable matrix factorization method to learn latent continuous representations of instances and logical formulas through low-rank approximation. The steps of ProPPR-MF are shown in [Figure 8: see original paper]. Through logical rule embedding learning, discrete logical facts and predicates are represented as continuous matrix vectors, improving reasoning efficiency and enabling the method to scale to large and complex knowledge graphs.

Traditional rule embedding learning methods adopt one-time injection of logical rules, ignoring the interactive characteristics between embedding learning and logical reasoning. These methods only focus on hard rules, which typically require substantial manual effort to create or verify. To overcome this problem, Guo et al. further proposed a soft rule-guided method called RUGE [71] based on KALE, using an iterative approach to replace one-round rule injection. RUGE uses AMIE to obtain rules but does not directly treat ground rules

as positive instances. Instead, it treats triples derived from rules as unlabeled triples and uses observable triples to update entity and relation embeddings. Since the truthfulness of unlabeled triples is difficult to determine, it predicts the probability of each unlabeled triple based on current embeddings. Finally, embeddings are updated based on both labeled and unlabeled triples. Triple embeddings adopt the ComplEx method [72], while rule embeddings use the same t-norm fuzzy logic method as KALE. The algorithm pseudocode for its iterative learning process is shown in Algorithm 1.

KALE and RUGE calculate rule scores based on the combination of component scores of rules or formulas. However, since triple scores are estimated separately, even if triples in a rule or formula are completely unrelated, they may still result in high rule or formula scores. To address this issue, Wang et al. [73] convert a triple or a ground rule into first-order logic, then score this first-order logic through vector or matrix operations on the embeddings of entities and relations contained in the first-order logic. illustrates the format of first-order logic, and introduces how to score first-order logic through mathematical expressions. Through this conversion, different triples contained in the same rule interact directly in vector space, ensuring a one-to-one mapping conversion for both rules and their encoding formats, making rule scoring more reasonable and reasoning accuracy higher.

Most embedding representation and rule-based methods ignore the transitivity and antisymmetry of logical rules. To address this problem, Wang et al. [74] proposed the TARE method, whose core idea is to combine knowledge graph triples, existing relations, and logical rules to learn knowledge graph embeddings, using the transitivity and antisymmetry of logical rules to approximately rank relation types in logical rules. First, it uses the ComplEx method to embed entities and relations, constrains relation vector embeddings to be non-negative, and reduces the problem to non-negative matrix factorization. The loss function for the embedding learning process is: $L = \sum_{(h,r,t)\in\Omega} \log(1 + \exp(-y_{hrt} \cdot \phi_{hrt}))$, where ϕ_{hrt} is the energy function in ComplEx, Θ represents model parameters, $y_{hrt} \in \{-1, 1\}$, and $\Omega \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of labels indicating whether triples are true or false. Then, it obtains the order between relations by ranking the real and imaginary parts of complex vector embeddings separately. For example, for rule $r_1(x, y) \wedge r_2(y, z) \Rightarrow r_3(x, z)$, the loss function for the ranking process is: $L_{rule} = \max(0, \lambda - (\mathbf{r}_3 - \mathbf{r}_2) \cdot \mathbf{r}_1)$, where P and N represent positive and negative sets of logical rules, α is a margin hyperparameter, F represents the penalty function score of logical rules, and R' is the set of negative rules constructed by replacing c_r in the result with a random relation $r' \in \mathcal{R}$. Finally, the loss functions L_{triple}, L_{rule}, and L_{rank} are jointly optimized using stochastic gradient descent. In the TARE method, logical rules are directly incorporated into relation type representations rather than instantiated with specific entities. Therefore, the learned embeddings are compatible not only with triples but also with rules, and relation type embeddings are approximately ranked.

Ding et al. [75] proposed the ComplEx-NNE-AER method, which incorporates

No-NEgative Constraints (NNE) on entity representations and Approximate Entailment constraints on Relation (AER) into the embedding model to add prior beliefs to the structure of embedding space without affecting embedding and transferability. NNE helps learn compact and interpretable representations of entities, while AER further encodes logical rules entailed between relations into their distributed representations. This method imposes three constraints for embedding representation learning based on ComplEx:

$\mathrm{Re}(\mathbf{e}) \geq 0, \mathrm{Im}(\mathbf{e}) \geq 0, \forall e \in \mathcal{E}\ ||\mathrm{Re}(\mathbf{e})||_2 \leq \alpha, ||\mathrm{Im}(\mathbf{e})||_2 \leq \beta, \forall e \in \mathcal{E}\ \mathbf{p}_r \geq \lambda \mathbf{q}_r, \forall (p \rightarrow q) \in \mathcal{R}$

where Re and Im represent the real and imaginary parts of embedding vectors, $p \rightarrow q$ indicates that relation p_r entails q_r with confidence λ, and α and β are slack variables. Equation 13 ensures that only positive attributes are stored in entity representations and keeps entity representations within the hypercube $[0, \alpha]^d$; Equation 13 ensures the model does not need to traverse all potential entity pairs and uses encoding with different confidence levels to make the model more robust to uncertainty. The ComplEx-NNE-AER method is simple, effective, and universally applicable. NNE and AER improve model efficiency and interpretability, adding structure to embedding space to make it applicable to large-scale knowledge graphs.

Niu et al. [76] combined rule learning, embedding representation, and path learning to propose a joint embedding method based on rules and paths called RPJE (Rule and Path-based Joint Embedding). First, it mines Horn rules of different lengths (number of relations in rule body) from knowledge graphs and encodes them for embedding representation. Then, it uses rules of length 2 to precisely compose paths and rules of length 1 to create semantic associations between relations, constraining relation embeddings. Additionally, the optimization process considers the confidence of each rule to ensure the applicability of rules to embedding representation. The model structure of RPJE is shown in [Figure 9: see original paper]. This method fully utilizes the interpretability and accuracy of logical rules, the generalization of knowledge graph embeddings, and the supplementary semantic structure of paths, effectively alleviating the problems of difficulty adapting embedding representation learning methods to sparse knowledge graphs and poor interpretability.

Additionally, Tang et al. proposed the RulE [77] method, which models logical rules and triples through embedding representation. RulE uniformly embeds entities, relations, and logical rules into a joint space for co-representation, then learns embeddings for each logical rule. This method performs logical rule inference in a soft manner and calculates confidence scores for each ground rule. Compared with pure embedding representation methods, RulE allows injecting prior logical rule information into embedding space, which improves the generalization ability of knowledge graph embeddings. Moreover, the learned rule confidence scores improve the logical rule inference process by softly controlling each rule's contribution, thereby alleviating the robustness problem of rules.

Rule embedding representation learning methods can improve the efficiency and interpretability of knowledge reasoning. However, since such methods essentially still perform embedding representation of knowledge graphs, the model itself remains uninterpretable. In other words, these methods only have result interpretability, not process interpretability.

5.2 Mutual Enhancement Between Embedding Representation and Rules

Rule learning and embedding learning are complementary. On one hand, rules can infer additional triples for sparse knowledge graphs, enabling more accurate embedding learning. On the other hand, embedding representation with rich semantic encoding can transform discrete graph search-based rule learning into vector space computation, significantly reducing search space. Therefore, methods with mutual enhancement between embedding representation and rules can balance efficiency improvement and reasoning process interpretability.

Zhang et al. [78] argued that existing rule learning-enhanced knowledge graph embedding methods only utilize existing rules and fail to use embedding representation to update rules during reasoning, which significantly impacts reasoning accuracy. To address this issue, they proposed the IterE method. IterE aims to simultaneously learn embeddings and rules, inferring new rules in each iteration based on updated embeddings. The IterE algorithm mainly includes three parts: (1) embedding learning; (2) axiom induction; (3) axiom injection. The embedding learning step mainly uses triples existing in the knowledge graph and triples inferred by rules to learn entity and relation embeddings. The axiom induction step uses an effective pruning strategy to generate an axiom pool, then scores each axiom in the pool by computing relation embeddings according to rule conclusions based on linear mapping assumptions. The axiom injection part mainly uses the deductive capability of axioms to infer new triples about sparse entities based on basic axioms, and injects these new triples into the knowledge graph to improve sparse entity embeddings. The model framework of IterE is shown in [Figure 10: see original paper]. IterE uses axioms to improve the quality of sparse embeddings, uses embeddings to improve the efficiency and quality of rule learning, and iterative training ensures better link prediction performance and high-quality rule learning results.

Due to the existence of missing facts in knowledge graphs, learning high-quality rules using only the information in the graph itself is difficult. To address this problem, Ho et al. [79] proposed a method called RuLES for learning high-quality rules with the help of embedding model feedback in the context of missing triples. This method starts from an initial rule, adds atoms to the rule body one by one, and uses embedding models trained on joint knowledge graphs and text corpora to score the quality of triples generated by rules for pruning and updating rules. The steps of the RuLES method are shown in [Figure 11: see original paper].

The model input of RuLES includes knowledge graph G, text corpus, and specified parameters for terminating rule construction, including rule learning, embedding learning, and rule evaluation modules. The rule learning module constructs rules based on the knowledge graph. The rule evaluation module uses G and the quality scoring function f given by the embedding model to provide quality scores μ for rule r (including rule description quality score μ_1 based on rules and prediction probability score μ_2 based on embedding). The embedding module uses G and text to compute what is called a "probabilistic" knowledge graph $\mathcal{P}(G)$. The model models rules as sequences of atoms, where the first atom is the rule head and other atoms are the rule body. In the initial stage, all possible binary atoms appearing in G are added to a rule queue with empty rule bodies. In each iteration, a rule is selected from the queue. If the rule satisfies filtering conditions, the system returns it as output; if not filtered, it uses refinement operations for rule expansion and adds new candidate rules to the queue. By comprehensively evaluating rule quality in incomplete graphs and incorporating embedding feedback on rule quality, RuLES can learn higher-quality rules for reasoning.

To improve reasoning accuracy on large-scale knowledge graphs with many predicates, Omran et al. [80] proposed the RLvLR method. This method combines embedding representation learning technology with a new sampling method to learn rules from large knowledge graphs. RLvLR mainly applies three techniques: (1) Subgraph sampling technology, which reduces the scale of input knowledge graphs by deleting triples with low relevance to target predicates, improving the transferability of embedding representation; (2) Parameter embedding technology for quality assessment of rules; (3) Rule quality evaluation technology based on matrix operations to avoid expensive query answering operations. First, RLvLR organizes entities and relations within path length N around target predicates into subgraphs. Then it searches the subgraphs to find paths related to predicates to discover rules, uses the RESCAL method [125] to generate embeddings for entities and predicates in rules, and designs scoring functions based on semantic similarity and predicate co-occurrence ideas to guide rule pruning, obtaining a candidate rule set. Finally, it performs final evaluation through efficient adjacency matrix multiplication to obtain a set of closed rules. Based on RLvLR, Orman et al. also proposed a new method called RLvLR-stream for learning rules from knowledge graph streams to complete temporal knowledge graph prediction. RLvLR shows outstanding performance in rule learning quality and efficiency, and its link prediction accuracy is also superior to similar methods. Algorithm 2 provides the specific process of the RLvLR algorithm.

In large-scale knowledge graphs, the number of ground rules is large, and mutual enhancement methods between embedding representation and rules such as IterE and RuLES require sampling ground rules to solve transferability problems, which greatly impacts reasoning efficiency. To address this issue, Cheng et al. [81] proposed the UniKER reasoning framework, which constrains logical rules to Horn rules and fully utilizes knowledge in rules to enable rule learning

and embedding representation to enhance each other in an efficient manner. To determine Horn rules, UniKER designs an improved forward chaining Embedding algorithm that adopts "lazy inference," only using a small portion of basic predicates and rules, and can activate more basic predicates and rules during inference, avoiding computation of large numbers of unused basic predicates and rules, greatly improving reasoning efficiency. Then, it updates the knowledge graph through embedding-based variational inference, using embedding representation methods to add useful hidden triples and delete incorrect triples from the knowledge graph and inference results. UniKER fully utilizes knowledge contained in rules to achieve better embeddings, while embedding representation enhances forward chaining by providing more potentially usable triples, enabling rule learning and embedding representation to mutually promote and jointly improve reasoning quality.

Mutual enhancement methods between embedding representation and rules aim at link prediction and rule learning, can learn higher-quality rules faster, and effectively enhance the interpretability of reasoning results. However, as the number of reasoning hops increases, prediction performance becomes more sensitive to knowledge graph sparsity, and reasoning efficiency also decreases. provides information on reasoning methods based on embedding representation and rules.

6 Knowledge Graph Reasoning Based on Neural Networks and Rules

In recent years, deep learning technology has achieved excellent performance in computer vision, natural language processing, and network link prediction tasks, and has been widely used in knowledge graph representation learning and reasoning. Deep learning methods can effectively model implicit features and correlations within data by establishing various neural network models, demonstrating superior performance in speed, accuracy, robustness, and generalization, but cannot provide explanations for reasoning results. Fusing neural networks and rules to achieve efficient and interpretable reasoning has become a research focus in knowledge graph reasoning. Based on whether neural networks or rules play a dominant role in reasoning, such methods can be divided into knowledge graph reasoning with rule-improved neural network modeling and knowledge graph reasoning with neural network-assisted rule learning.

6.1 Rule-Improved Neural Network Modeling for Knowledge Graph Reasoning

Rule-improved neural network modeling for knowledge graph reasoning focuses on how to use logical rules to enhance the embedding and prediction capabilities of neural networks. In these methods, neural networks not only learn entity and relation embeddings on original observed triples in knowledge graphs but also learn triples or ground rules inferred from some predefined rules. Rules can

guide the training process of neural networks, and added triples can serve as augmentation of training datasets, making results from neural network-based knowledge graph reasoning more interpretable and improving model efficiency.

Wang et al. [82] proposed the Logic Attention Network (LAN), which performs entity embedding by aggregating neighbors based on attention weights from both rules and networks. They first proposed that an ideal domain information aggregator possesses three properties: permutation invariance, redundancy awareness, and query relation awareness. Based on this idea, they designed LAN. LAN uses an RNN [88]-based Encoder-Decoder as the basic framework and employs both logical rule mechanisms and neural network mechanisms to jointly learn neighbor attention. The logical rule mechanism estimates weights at a coarse-grained relation level, quantifying potential dependencies between rules as confidence $\alpha(r|q)$. If r is an important neighbor of q, r should satisfy $\alpha(r|q) > \alpha(r'|q)$, where r' represents other relations besides r and q. Then, logical attention weights are defined as:

$$\alpha_{j|i,q}^{\text{Logic}} = \frac{\exp(\alpha(r_{ij}|q))}{\sum_{k \in N_i} \exp(\alpha(r_{ik}|q))}$$

By calculating $\alpha_{j|i,q}^{\text{Logic}}$, the logical rule mechanism addresses query relation awareness and neighborhood redundancy awareness. Simultaneously, a neural network mechanism is introduced to represent neighbor weights at a fine-grained level, calculating importance weights $\alpha_{j|i,q}^{\text{NN}}$ after describing neighbor embeddings. Finally, the two mechanisms are weighted and summed to describe neighbor entity importance:

$$\alpha_{j|i,q} = \lambda \cdot \alpha_{j|i,q}^{\text{Logic}} + (1 - \lambda) \cdot \alpha_{j|i,q}^{\text{NN}}$$

LAN's entity representation has stronger learning ability, showing significant improvements in reasoning efficiency, accuracy, and interpretability.

Sun et al. [83] proposed GRAFT-Nets (Graphs of Relations Among Facts and Text Networks), a method for incomplete knowledge graph reasoning with textual rule information in open-domain question answering tasks. This method has two main innovations: first, it proposes heterogeneous update rules for knowledge graph nodes and text nodes, building LSTM [84] to propagate information between text rule nodes; second, drawing on the idea of the Personalized PageRank (PPR) [85] algorithm, it proposes a directed propagation method to constrain the direction of graph embedding propagation, keeping it always on paths starting from question seed nodes. First, for query questions, LSTM is used to initialize latent representations of text rules. Then, feedforward networks are used to update entity and text rule nodes separately. Finally, attention relations and personalized propagation are used to introduce dependencies with query questions, with the former ensuring embeddings propagate more along question-related edges and the latter enabling high weights along question-related paths. GRAFT-Nets can effectively handle heterogeneous graphs of knowledge graphs and text, extracting answers from specific question subgraphs containing text rule information and entities and relations.

Most embedding representation learning methods assume all test entities are available during training, making retraining embeddings very time-consuming when new entities appear. To address this problem, He et al. [86] proposed a graph neural network-based embedding representation framework called the Virtual Neighbor (VN) network. The VN network focuses on solving three challenges: first, to alleviate neighbor sparsity, it introduces the concept of virtual neighbors inferred by rules, i.e., assigning soft labels to neighbors through constraint rules rather than treating them as fixed and unchanging; second, many existing methods only use single-hop or two-hop neighbors for aggregation, ignoring potentially helpful multi-hop information, while the VN network uses graph convolutional networks to simultaneously identify logical and symmetric path rules to capture complex patterns; third, it adopts an iterative learning method to capture interactions between embedding representation and virtual neighbor predictions for updated rule learning. [Figure 12: see original paper] shows the framework of the VN network. The framework mainly includes three parts: (1) rule-based virtual neighbor prediction; (2) an encoder based on Weighted Graph Convolutional Network (WGCN) [87] to capture structural information and embed entities; (3) a decoder that calculates edge probabilities and uses soft labels to refine the model. The VN network not only has higher reasoning accuracy but also significantly stronger robustness to neighbor sparsity.

MLN can utilize domain knowledge from first-order logic while handling uncertainty, but it struggles to adapt to complex graph structures and is difficult to reason with. Neural network modeling has high reasoning efficiency but cannot utilize domain knowledge. Qu et al. [89] proposed pLogicNet, a probabilistic logic neural network that combines the advantages of MLN and neural network embedding representation learning. pLogicNet uses a Markov logic network with first-order logic to define the joint distribution of all possible triples and efficiently optimizes it using a variational EM algorithm. In the E-step, it parameterizes the variational distribution as a knowledge graph embedding model and uses amortized mean-field [90] inference to calculate the likelihood probability of unobserved triples for reasoning about unobserved triples. In the M-step, it calculates pseudo-likelihood based on observed triples and predicted triples to update the weights of logical rules, finally using gradient descent for training.

Similarly, Vardhan et al. [92] proposed pGAT (probabilistic Graph Attention neTwork), a probabilistic logic graph attention network based on MLN and graph attention networks [91]. pGAT uses a variational EM algorithm to optimize the joint distribution of all possible triples defined by MLN, effectively combining first-order logic and graph attention networks. In the E-step, GAT embeddings are used to infer unobserved triples; in the M-step, weights of Markov logic network rules are updated based on observed triples and inferred triples obtained from the above embeddings. The framework of the pGAT method is shown in [Figure 13: see original paper]. pGAT combines the probabilistic logical rule expression capability of MLN with the ability of graph attention networks to capture adjacent entity features and adapt to highly complex graph problems, enabling application to large-scale complex knowledge graph reason-

ing.

Additionally, Zhang et al. [93] also studied how to combine Graph Neural Networks (GNN) with MLN, proposing the ExpressGNN method. The essence of ExpressGNN is an inference network for MLN that scales MLN reasoning to larger knowledge graph reasoning problems by leveraging prior knowledge encoded in logical rules and structural knowledge implicitly encoded in knowledge graphs captured by GNN. [Figure 14: see original paper] shows the model framework of ExpressGNN. First, it models the joint probability distribution of all observed and latent variables using MLN, solved by optimizing the variational evidence lower bound (ELBO) of data log-likelihood. Then, in the E-step, it uses a mean-field distribution to approximate the true posterior distribution and uses GNN to parameterize the variational posterior. Finally, in the M-step, it only retains ground predicates to compute pseudo-likelihood and learn the weights of logical formulas in MLN with fixed variational posterior. ExpressGNN not only has higher reasoning accuracy but can also trade off model compactness and expressiveness by adjusting the dimensions of GNN and embedding parts. Moreover, ExpressGNN can handle few-shot learning problems with few target predicates or zero-shot learning problems with zero labeled instances.

Furthermore, some scholars use rules as auxiliary knowledge to guide reinforcement learning agents in path exploration for knowledge graph reasoning. For example, Lei et al. [94] proposed the RuleGuider method, which first uses high-quality rules generated by the AnyBURL method to provide reward supervision for agents, then decomposes the agent into two sub-agents based on the structure of rules and their entity variables: an entity agent and a relation agent. The entity agent generates entity distributions for selectable entities; after the relation agent samples a relation, it prunes the entity space based on the selected relation. The entity agent then samples entities based on the pruned entity space. In the final step, they obtain hit rewards based on the last selected entity and rule guidance rewards from the pre-mined rule set based on the selected relation path.

Xia et al. [95] proposed SparKGR, a hybrid multi-hop reasoning method with reinforcement learning for sparse knowledge graph reasoning. This method uses rule guidance to dynamically complete missing paths to increase the action space of reinforcement learning agents, alleviating the incompleteness and sparsity of knowledge graphs. It then designs an iterative optimization method for rule induction and fact reasoning to combine global information from knowledge graphs to guide agent exploration. SparKGR can effectively reduce knowledge graph sparsity, improve path exploration efficiency, and effectively improve reasoning accuracy while ensuring interpretability. Agents use three strategies to select the next action space: dynamic path completion, iterative rule guidance, and policy function guidance. Finally, the probability distributions of the three strategies are weighted and summed to determine the agent's next action. [Figure 15: see original paper] presents SparKGR's dynamic path completion strategy and iterative rule guidance strategy.

Methods using rules to improve neural network modeling can provide data assistance and direction guidance for neural network training and parameter learning, making reasoning results somewhat interpretable. However, these methods do not reveal patterns from internal model structures and parameters, failing to fundamentally change the "black-box" nature of neural networks. Methods combining rules and deep reinforcement learning have strong interpretability, but their interpretability comes from agents' path exploration rather than the neural network itself.

6.2 Neural Network-Assisted Rule Learning for Knowledge Graph Reasoning

These methods leverage the advantages of neural networks to handle data uncertainty and fuzziness, narrowing the search space for rule reasoning to discover higher-quality rules for knowledge graph reasoning. They mainly include neural theorem proving-based methods, matrix multiplication-based methods, and neural inductive learning-based methods.

(1) Neural Theorem Proving-Based Methods

Neural theorem proving-based methods are based on the Prolog programming language [96]. Prolog is a deductive reasoning logic programming language containing steps such as predicate matching, rule solving, and backtracking discrimination. It performs knowledge reasoning by combining recursive matching algorithms with neural network embedding representation and provides interpretable theorem proving paths.

Rocktäschel et al. [97] combined Prolog and ILP to propose an end-to-end knowledge reasoning method called NTP (Neural Theorem Prover). NTP draws on Prolog's backward chaining algorithm to recursively build neural networks operating on dense vector representations of symbols, then uses a Differentiable Prover (DP) to perform end-to-end differentiable proving of knowledge graph reasoning. NTP takes atoms, rules, and their proof states as input and returns a list of new proof states. First, it uses three modules of DP to prove queries on knowledge graphs: (1) Unification module, which uses radial basis function kernels for differentiable computation of vector representations of rule atoms instead of symbolic unification, combining rule reasoning with learning vector representations of atoms and rules; (2) OR module, which applies rules in knowledge graphs; (3) AND module, which verifies success or failure and recursively works with the OR module for proof states and sub-goals. Then, it calculates the proof success score of the target by proof aggregation. Finally, it uses gradient descent algorithms to apply NTP to ILP, replacing traditional combinatorial search strategies to train neural networks and learn interpretable rules from data with the help of backpropagation of prediction errors. NTP can infer facts from given incomplete knowledge graphs and introduces interpretable first-order logic rules. However, this method's execution process requires enumerating and evaluating all bounded-depth proof paths for given targets, making NTP inefficient

in large knowledge graphs.

To address NTP's difficulty in applying to large knowledge graphs, Minervini et al. [98] proposed an improved method called GNTP (Greedy NTP) that performs joint reasoning on knowledge bases and text by embedding logical facts and natural language sentences in a shared embedding space. To handle NTP's problems, GNTP uses three techniques: (1) Use fact embeddings to select nearest neighbor facts for proving sub-goals and relation embeddings to select rules for expansion, thereby only considering a subset of proof paths related to maximum proof scores, reducing the number of candidate proof paths and spatial complexity; (2) Introduce attention mechanisms for rule induction to handle known predicates, improving model parameter efficiency and reducing time complexity; (3) Extend NTP to natural language, using an end-to-end differentiable reading component to jointly embed predicates and text surface patterns in a shared space to fully utilize natural language text information and provide more knowledge for reasoning. GNTP maintains the interpretability of NTP while alleviating its complexity and transferability limitations, enabling application to real-world large-scale knowledge graphs.

NTP needs to consider all rules that explain given targets or sub-goals during training, leading to significantly reduced efficiency in cases with many rules or reasoning steps. To narrow the rule search space, Minervini et al. [99] proposed the CTP (Conditional Theorem Prover) method. This method dynamically generates minimal rule sets by building neural networks and uses gradient-based optimization to learn optimal rule selection strategies. For each sub-goal, CTP's selection module first dynamically generates a minimal rule set for use in subsequent reasoning steps. Then, it proposes three restructuring methods for rule selection: neural goal reformulation, attention-based goal reformulation, and memory-based goal reformulation. Among them, neural goal reformulation defines the "selection" operation as a linear function of target predicates, generating a smaller set of rules conditioned on sub-goals that can also be trained end-to-end in downstream reasoning tasks. Attention-based goal reformulation generates attention distributions over relation sets based on predicates to incorporate useful priors in the selection module. The memory-based goal reformulation module stores rules in a differentiable memory that can produce attention distributions over rules given targets for key-value memory lookup. Compared with NTP, CTP has stronger transferability and higher reasoning accuracy.

The disadvantage of such methods is that theorem proving often requires computationally intensive rule search, making it difficult to balance the precision and efficiency of rule search. Moreover, neural theorem proving methods themselves are deductive reasoning methods that cannot discover new rules.

(2) Matrix Multiplication-Based Methods

These methods define relations as two-dimensional matrices, use matrix multiplication to represent multi-hop reasoning, and use recurrent neural networks to simulate rule reasoning. Matrix multiplication is differentiable, so models

can achieve synchronous learning of rule parameters and rule structures to handle complex knowledge graph reasoning tasks. Moreover, the reasoning process is obtained through layer-by-layer operations of matrix multiplication at each level, providing clearer explanatory basis for reasoning results.

Cohen proposed the probabilistic deductive data framework TensorLog [100], which can infer weighted chain logical rules to explain relations in knowledge graphs for knowledge reasoning. In TensorLog, each entity in the knowledge graph is represented by one-hot embedding, and each relation R is represented as a matrix M_R, where $M_R[i,j] = 1$ if $R(e_i, e_j) \in \mathcal{F}$. Given head entity X, answer Y can be logically reasoned by performing matrix multiplication $M_{R_n} \times ... \times M_{R_1} \times v_X$ to obtain v_Y. Since query relations can be explained by multiple rules, the score of query relations is calculated by combining all rules:

$$P(R|X) = \sum_{\gamma \in \Gamma} \alpha_\gamma \prod_{R_i \in \beta_\gamma} M_{R_i} v_X$$

where Γ represents all possible rules, α_γ is the confidence related to γ, and β_γ is the ordered list of all predicates in γ. During reasoning, given head entity X, the probability of each correct answer Y satisfying $P(R|X)$ is maximized:

$$\max_\Theta \sum_{(X,R,Y) \in \mathcal{F}} \log P(Y|X,R)$$

TensorLog first converts each clause in logical rules into a factor graph, where each node represents a variable in the rule and each edge represents a predicate or relation in the rule. Then, for different types of queries on the factor graph, it performs Belief Propagation (BP), and the required message passing steps are "unrolled" into a differentiable function to calculate probabilities. Finally, recursively combining these differentiable functions enables reasoning in non-trivial logical theories containing multiple interrelated rules and predicates.

TensorLog estimates rule confidence through differentiable matrix generation but cannot generate new logical rules. Moreover, since each rule is associated with a parameter and enumerating rules has high computational complexity, parameter learning is very difficult. To address TensorLog's shortcomings, Yang et al. [101] proposed Neural Logic Programming (Neural LP), improving TensorLog's matrix reasoning framework. Neural LP swaps the sum and product in Equation 17 and decomposes rule weights into weights of predicates in rules. To handle rules of different lengths, Neural LP also designs a recursive method for dynamic rule modeling:

$$u^{(t+1)} = \sum_{r \in \mathcal{R}} a_t[r] \cdot M_r u^{(t)}$$

for $t = 1, ..., T$, where T is the maximum rule length, R is the number of predicates in the knowledge graph, and u is the auxiliary storage vector of input entity X_v. In each recursion, it first uses memory attention vector b_t to compute weighted averages of previous auxiliary storage vectors, then uses operation attention vector a_t to perform TensorLog operations. Finally, it computes weighted averages of all auxiliary storage vectors and uses attention to control rule length. It uses RNN modeling to solve for a_t and b_t and uses them as basis

for reconstructing rules and their confidence.

Neural LP is limited in handling numerical features such as age, weight, and scientific measurements, restricting the types of rules it can represent. To address this, Wang et al. [102] proposed Num-Neural LP to learn numerical rules. Based on Neural LP, this method expresses numerical comparator operators using dynamic programming and cumulative sum operations, and jointly uses them with sparse operators to adapt to dense operations. Additionally, Num-Neural LP adds two operators: a classification operator for comparing functions on entity attributes, and a negation operator that, based on the local closed-world assumption, flips elements in rows containing at least one non-zero element for a given negation operator. Num-Neural LP can aggregate and extract more expressive rules, with learned rules having higher quality and more accurate reasoning results.

To mine variable-length first-order logic rules from knowledge graphs, Sadeghian et al. [103] proposed the end-to-end differentiable model DRUM. DRUM can simultaneously learn rule structures and confidence scores, achieving inductive and interpretable knowledge graph reasoning. First, it converts rule reasoning into vector multiplication calculations. Then, it introduces a special relation B_0 with adjacency matrix as identity matrix I to learn variable-length rules, and introduces confidence tensors to avoid learning high-confidence wrong rules. Finally, it uses Bi-RNN's ability to capture sequential information of atoms in rules to model associations between relations in rule heads and bodies. In addition to advantages in reasoning accuracy, DRUM can also perform inductive reasoning, handle unseen entities, and has strong interpretability.

Limited by the properties of matrix multiplication, most methods such as TensorLog, Neural LP, and Num-Neural LP can only learn chain logical rules and cannot handle complex rule forms such as tree-like and conjunctive rules. Inferring rules based on specific head entities affects the generalization ability of such methods. Moreover, since the conversion of entities, relations, and rules to matrices in these methods depends on embedding representation methods, the robustness of the models themselves is not strong.

(3) Neural Inductive Learning-Based Methods

These methods use graph neural networks and recurrent neural networks for inductive logic rule learning and then perform reasoning prediction. Graph neural networks use graph data as representation methods, can perform optimized calculations on nodes and relations, and consider nodes' hidden information, neighbor node information, and graph structure information, making them very suitable for rule learning in knowledge graphs. Recurrent neural networks excel at modeling data with sequential information and can also be applied to knowledge graph reasoning. Neural inductive learning has become a research focus for scholars in recent years.

Yang et al. [104] proposed the Neural Logic Inductive Learning (NLIL) method. NLIL handles non-chain rules by combining primitive statements. Primitive

statements refer to predicates applicable to logical variables or results of certain operators. Unary primitive statements contain relation paths starting from a variable, binary primitive statements contain two relation paths starting from two different variables, and primitive statements can also be logically combined through operators $\{\wedge, \vee, \neg\}$, giving them the ability to represent tree-like and conjunctive logic rules, as shown in [Figure 16: see original paper]. In rule learning, to determine the importance weights of different primitive statement logic combinations, it uses three stacked Transformer [105] networks to learn different attention vectors.

Dong et al. [106] proposed a neuro-symbolic architecture for inductive learning and logical reasoning called Neural Logic Machines (NLM). NLM uses neural networks as function approximators and employs logic programming to handle objects with properties, relations, logical connectives, and quantifiers. Internally, NLM uses tensors to represent logical predicates by grounding predicates to True or False on a fixed set of objects. Then, it uses Multi-Layer Perceptron (MLP) to build a neural architecture for learning rules, using Boolean logic and quantification methods such as expansion and reduction to represent meta-rules in the architecture. After training on small-scale tasks, NLM can recover improved rules and generalize to large-scale tasks.

Chen et al. [107] proposed a GNN-based Graph Collaborative Reasoning (GCR) method that performs relation reasoning on knowledge graphs using adjacent link information from a logical reasoning perspective, and uses GNN's message passing function to aggregate rich information from adjacent links for efficient reasoning. First, it converts graph structures into logical expressions of Horn rules, thereby transforming knowledge reasoning tasks into neural logical reasoning problems. Then, it encodes each triple as predicate embeddings and models each relation as a neural module to encode triples. Finally, based on encoded predicate embeddings, it builds network structures using neural modules according to modeled Horn rules and uses backpropagation to learn model parameters. GCR can fully utilize logical relationships between links, does not require manually predefined rules, has strong interpretability and transferability, and is particularly suitable for knowledge graph reasoning with sparse data.

To achieve efficient inductive reasoning in knowledge graphs with incomplete training sets, Teru et al. [108] proposed a relation prediction framework based on graph neural networks called GraIL (Graph Inductive Learning). GraIL can mine first-order logic rules by learning subgraph embedding representations to predict relations from subgraph structures around candidate relations. GraIL mainly includes three steps: (1) Subgraph extraction, which extracts closed subgraphs around target nodes to mine logical evidence needed to derive relations between target nodes, explicitly encoding rules; (2) Node labeling, which uses distances between nodes and surrounding subgraph nodes to build node feature matrices as input to GNN to initialize neural message passing algorithms; (3) GNN scoring, which uses GNN to score the possibility of triples. Using neural message algorithms, node representations are iteratively updated by combining

(combine) node representations with aggregation (aggregate) of neighbor representations. For example, for the k-th layer of GNN, the iterative update formula for node representations is:

$$\mathbf{h}_t^{(k+1)} = \text{COMBINE}^{(k)}(\mathbf{h}_t^{(k)}, \mathbf{a}_t^{(k)})$$

where $\mathbf{a}_t^{(k)} = \text{AGGREGATE}^{(k)}(\{\mathbf{h}_s^{(k)} | s \in \mathcal{N}(t)\})$ represents aggregated messages from neighbors, $\mathbf{h}_t^{(k)}$ is the latent representation of node t at layer k, and $\mathcal{N}(t)$ is the direct neighbor set of node t. The AGGREGATE function is defined based on multi-relational R-GCN [126] and edge attention, while the COMBINE function is determined by the ReLU function of R-GCN. Finally, it uses subgraph representation $\mathbf{h}_u^{(L)} \oplus \mathbf{h}_v^{(L)}$ and target relation embedding representation $\mathbf{r}$ to calculate the possibility score of triple (u, r, v), i.e.:

$$\text{score}(u, r, v) = \text{MLP}(\mathbf{h}_u^{(L)} \oplus \mathbf{r} \oplus \mathbf{h}_v^{(L)})$$

Reasoning is performed based on the magnitude of possibility scores. [Figure 17: see original paper] shows the process steps of the GraIL method. GraIL can learn useful subsets of first-order logic rules, reason about entities not present in the training set, and has stronger accuracy and interpretability compared to other inductive reasoning methods.

GraIL initiated the research on using graph neural networks for inductive logic rule learning. Subsequently, researchers conducted multiple related improvement works. To address GraIL's problem of ignoring the directional nature of knowledge graphs when extracting closed subgraphs of target triples, making it unable to handle asymmetric and antisymmetric relations, Mai et al. [109] proposed the CoMPILE method based on GraIL, using node-edge communication message passing mechanisms to replace the original GNN to measure relation importance. This method first extracts directed closed subgraphs for triples, then expands the communication message passing network framework to strengthen information interaction between entities and relations, updating edge and entity embeddings simultaneously. Finally, it uses edge-aware attention mechanisms to aggregate local neighborhood features and collect global entity information to enrich entity and relation representations. Compared with GraIL, CoMPILE can perform communication message passing in directed subgraphs to handle asymmetric and antisymmetric relations while controlling model parameter numbers.

Additionally, some inductive knowledge logic rule learning methods are not based on GraIL improvements but have performance improvements over GraIL. For example, to address GraIL's problem of not considering semantic relevance, Chen et al. [110] proposed the TACT (Topology-Aware CorrelaTions) method, which considers correlations between relations in subgraphs, encodes relation correlation networks to enhance encoding of closed subgraphs, and performs graph topology-aware correction. To address the difficulty of balancing entity-independent relation modeling needs and discrete logical reasoning interpretability, Lin et al. [111] proposed the ConGLR method, which combines context

graphs and logical reasoning. Based on extracting closed subgraphs of target head and tail entities, it builds context graphs containing relation paths, relations, and entities, uses attention-aware graph convolutional networks to process subgraphs and context graphs, and finally calculates confidence scores by combining neural computation and logical reasoning with relation paths as rule bodies and target relations as rule heads. To address the problem of difficult-to-handle sparse subgraphs, Xu et al. [112] proposed the SNRI (Subgraph Neighboring Relations Infomax) method, which uses neighboring relation features and neighboring relation paths to capture complete neighboring relations of entities in subgraphs and globally models neighboring relation paths through mutual information maximization, thereby merging complete relation information into closed subgraphs and modeling neighboring relation paths to improve inductive reasoning prediction performance. Zha et al. [113] proposed BERTRL (BERT-based Relational Learning), which combines pre-trained language models with relation learning. It first linearizes the knowledge graph as effective input for BERT, then builds a BERT model to encode reasoning paths, and finally uses a bag scoring method to aggregate scores of triple paths and reasoning paths. The framework of BERTRL is shown in [Figure 18: see original paper].

To obtain entity-related relation semantics from latent rules and solve the supervision insufficiency problem caused by rule scarcity in subgraphs, Pan et al. [114] proposed the RPC-IR (Relational Path Contrast for Inductive Reasoning) method based on graph convolutional networks. RPC-IR includes three steps: (1) Extract paths from closed subgraphs of target triples and generate positive and negative samples of relation paths for contrast; (2) Use GCN to obtain embedding representations of positive and negative samples; (3) Score target triples using subgraphs and relation paths and jointly train using a strategy considering both supervised and contrastive information. Through these steps, rules are determined based on the confidence of relation paths.

GraIL and its improved methods treat rules as paths, while path mining in large-scale graphs often faces huge search spaces. To address this problem, Yan et al. [115] proposed the CBGNN (Cycle Basis Graph Neural Network) method, which views logical rules as cycles from an algebraic topology perspective, builds GNN models, uses their message passing function to run implicit algebraic operations in cycle space to learn cycle representations, then learns rules by exploring the linear structure of cycle space, and performs knowledge reasoning. [Figure 19: see original paper] shows the model structure of CBGNN. The method mainly includes two stages. In the first stage, it builds a cycle basis and constructs cycle graphs for each cycle basis. In the cycle graph, nodes represent cycles in the basis, and nodes are considered connected if the corresponding cycles have strong interactions. Cycle information can be converted into node features of the new graph in the second stage. In the second stage, it builds GNN on the cycle graph to learn cycle confidence values and maps them to confidence values of target triples. In this stage, different cycle bases build different GNNs, but these GNNs share weights, and triple confidence can be calculated by aggregating them. Compared with rule path-based methods, CBGNN has

higher efficiency and provides inspiration for incorporating advanced topological information into graph representation learning.

To address the reasoning efficiency problems caused by large search spaces in neural logic programming methods and sparse rewards in reinforcement learning methods, Qu et al. [116] proposed the RNNLogic model. RNNLogic treats logical rules as a latent variable and trains a rule generator and reasoning predictor with logical rules. The rule generator is parameterized using RNN. During model optimization, it adopts an EM-based optimization algorithm. In each iteration, it first updates the reasoning predictor to explore logical reasoning rules. Then, in the E-step, it selects a set of high-quality rules from all generated rules through posterior inference; in the M-step, it uses rules selected in the E-step to update the rule generator. The model framework of RNNLogic is shown in [Figure 20: see original paper]. Here, q represents the query, z represents the latent representation of rules, θ and w represent model parameters, $p_\theta(z|q)$ represents the prior probability of generating rule latent distributions from queries, and $p_w(y|z,q)$ represents the probability likelihood of answers determined based on knowledge graphs, rule latent representations, and queries.

Additionally, to address few-shot knowledge graph reasoning problems, Huang et al. [117] proposed the CSR (Connection Subgraph Reasoner) method, which transforms few-shot knowledge reasoning into an inductive reasoning problem. This method does not require a pre-training stage and can directly predict target few-shot tasks. CSR designs a novel GNN-based encoder-decoder architecture to model shared connection subgraphs between support and query triples, and designs a self-supervised pre-training scheme to reconstruct automatically sampled connection subgraphs.

To address one-shot learning problems in knowledge graph reasoning, Du et al. [118] proposed the CogKR (Cognitive Knowledge Graph Reasoning) method, which includes summarization and reasoning modules. The summarization module summarizes basic relations of given instances, while the reasoning module performs reasoning based on relations summarized by the summarization module. In the summarization module, entity pair embeddings generated by GNN are used to obtain relations between entity pairs. In the reasoning module, CogKR draws on dual-process theory in cognitive science to build two systems for constructing cognitive graphs to store retrieved information and reasoning results. System 1 is an iterative coordinated retrieval system that intuitively collects reasoning-related evidence, and System 2 is a reasoning system that performs relation reasoning on collected information. The structural information of cognitive graphs enables the model to aggregate evidence from multiple reasoning paths and provide reasoning process explanations in graph form. Finally, reinforcement learning methods are used to transform graph structure modeling into policy optimization problems, and the REINFORCE algorithm [124] is used for policy optimization.

Overall, these methods have relatively high efficiency and accuracy and possess certain interpretability. However, since they use neural networks to learn rules,

they also introduce some uninterpretable factors to a certain extent. provides information on reasoning methods based on neural networks and rules.

7 Comparison and Analysis

The development from inductive logic programming-based reasoning methods, probabilistic graphical model and rule-based reasoning methods, embedding representation and rule-based reasoning methods to neural network and rule-based reasoning methods reflects continuous improvement and refinement of rule-based knowledge graph reasoning methods. Overall, the performance of the four categories of methods is progressively improving, but different categories and sub-methods have their own advantages and disadvantages in terms of efficiency, interpretability, and transferability.

Reasoning Accuracy. The main evaluation metrics for knowledge graph reasoning include Mean Reciprocal Rank (MRR), Hit@k (proportion of correct predictions in top k results, k=1,3,10), and AUC-PR (Area Under the Precision-Recall Curve). Equations 21 and 22 give the calculation formulas for MRR and Hit@k:

$$\text{MRR} = \frac{1}{|\mathcal{T}_{\text{test}}|} \sum_{(h,r,t)\in\mathcal{T}_{\text{test}}} \frac{1}{\text{rank}_h + \text{rank}_t}$$

$$\text{Hit@k} = \frac{1}{|\mathcal{T}_{\text{test}}|} \sum_{(h,r,t)\in\mathcal{T}_{\text{test}}} \mathbb{I}[\text{rank}_h \leq k \vee \text{rank}_t \leq k]$$

where $\mathcal{T}_{\text{test}}$ represents the number of facts in the test set, rank_h and rank_t represent rankings for head entity reasoning and tail entity reasoning respectively, and $\mathbb{I}[P]$ is an indicator function that returns 1 if condition P holds, otherwise 0. Based on experimental results from references, we summarize the experimental results of methods introduced in this paper on the classic knowledge graph reasoning dataset FB15K-237 [121] for link prediction tasks, as shown in . Methods that did not use relevant datasets, did not adopt the above metrics, performed dataset partitioning in experiments, or only targeted specific tasks (such as entity prediction) are left blank. Those only tested on the FB15K dataset (without removing inverse relations) are marked with a horizontal line below corresponding metrics, and those using few-shot subsets of datasets are marked with a wavy line below corresponding metrics. Some early methods were not tested on benchmark datasets and are difficult to reproduce in subsequent research, so experimental data could not be provided. As can be seen from , the accuracy of neural network and rule-based knowledge graph reasoning is generally higher than the other three categories of methods.

Reasoning Efficiency. In inductive logic programming-based reasoning methods, the computational complexity of first-order inductive learning methods increases exponentially with the number of rules, making them inefficient in medium-to-large-scale knowledge graphs. Association rule mining methods are difficult to apply to large knowledge graphs due to the need for rule traversal. Relation path sampling methods show significant efficiency improvements over the first two categories and can be used to learn rules in large-scale knowledge

graphs for reasoning. Probabilistic graphical model and rule-based reasoning methods have low overall efficiency due to complex search and probability calculations, making them mostly difficult to apply to large-scale knowledge graphs. Embedding representation and rule-based reasoning methods use representation learning to alleviate time-consuming processes such as search, traversal, and sampling, thus having higher reasoning efficiency than the first two categories. Among them, rule embedding representation learning methods have higher efficiency than mutual enhancement methods due to different dominant technologies. Finally, neural network and rule-based reasoning methods have significantly better efficiency than the first three categories.

Interpretability. From the perspective of interpretability evaluation, inductive logic programming-based reasoning methods have the strongest interpretability. Probabilistic graphical model and rule-based reasoning methods have relatively weaker interpretability due to probability estimation and fuzzy reasoning. Embedding representation and rule-based reasoning methods and neural network and rule-based reasoning methods introduce uninterpretable "black-box" models, affecting interpretability to some extent. Among these two categories, methods relying on rule learning for reasoning have higher interpretability than those relying on neural network modeling or embedding representation for reasoning.

Transferability. Transferability refers to the ability of models or methods to use existing knowledge to learn new knowledge. In knowledge graph reasoning, we understand transferability as the additional cost and performance difference when methods are applied to different knowledge graphs. Overall, benefiting from the flexibility and adaptability advantages of deep networks, neural network and rule-based reasoning methods have the strongest transferability, followed by embedding representation and rule-based reasoning methods. Inductive logic programming-based reasoning methods and probabilistic graphical model and rule-based reasoning methods have relatively poorer transferability because these methods need to relearn as many rules as possible when facing different knowledge graphs.

8 Challenges and Outlook

By combining with methods and models such as embedding representation and neural networks, rule-based knowledge graph reasoning methods have gradually improved past issues such as slow reasoning speed, high computational complexity, and limited expression ability, demonstrating good performance in large-scale knowledge graph reasoning. However, although rule-based methods have relatively strong interpretability, reliability, and generalization ability, there is still a gap in reasoning accuracy and efficiency compared to some state-of-the-art pure neural network-based reasoning methods. In future research, such reasoning methods still face many new challenges that need to be addressed, mainly including the following seven aspects:

Rule Learning Efficiency. Learning rules from large-scale knowledge graphs

is difficult. Although researchers have greatly improved rule learning efficiency by combining machine learning, deep learning, and reinforcement learning techniques, currently most rule learning methods rely on path search in knowledge graphs. The number of paths increases exponentially with the number of entities and relations. Therefore, research is needed on effective sampling strategies to reduce path search volume, or building pre-trained models to reduce search complexity by fusing prior information, text information, and corpora. At the same time, how to represent rules in forms more easily accepted by neural networks to further improve rule learning and reasoning effects also needs further exploration.

Rule Quality Assessment. Early rule quality assessment mainly calculated metrics such as confidence and support under the closed-world assumption, but these metrics are difficult to compute under data missing conditions. Later, Galárraga et al. [47] proposed assumptions such as Partial Completeness Assumption (PCA) to alleviate data missing problems by relaxing requirements for negative instances in confidence calculation formulas, but this assumption is only a compromise solution. Therefore, designing more reasonable and feasible rule evaluation metrics needs attention in future work. On the other hand, the purpose of rule learning is reasoning, and how rule quality affects knowledge graph reasoning accuracy is also worth systematic and in-depth research, because sometimes wrong rules may also infer correct answers.

Interpretability. The biggest advantage of rule-based knowledge graph reasoning methods is their interpretability, which gives them safety and reliability in practical applications. Embedding representation or neural modeling-dominated reasoning methods with rule representation as assistance have better efficiency than reasoning methods using neural networks and embedding representation to assist rule learning, but they lack interpretability. Neural network and embedding representation-assisted rule learning methods convert rules into vector space for reasoning, but these intermediate representations of vectors or matrices can only achieve true "model interpretability" when correctly explained. Future attempts can embed reasoning steps into models to mine and understand intermediate representations of rules. On the other hand, interpretable machine learning methods can be integrated with rules and embedding representation to improve efficiency as much as possible while ensuring interpretability.

Benchmark Datasets. Current commonly used datasets in knowledge graph reasoning, such as FB15K-237, NELL-995 [122], and WN18R [123], evaluate and compare the performance of supervised knowledge graph reasoning methods by dividing training, test, and validation sets. The performance of most current rule-based knowledge graph reasoning methods is also judged on the above datasets. However, these datasets cannot fully and accurately express all intrinsic rules and logical patterns. Some incorrect or redundant rules may also derive correct results, which can have a misleading impact on result interpretability. If applied in some important fields, it will inevitably lead to serious

consequences. Therefore, future benchmark datasets containing different predefined rules or logical patterns should be proposed to simultaneously satisfy the verification of rule learning and knowledge graph reasoning performance.

Temporal and Multi-Modal Knowledge Graph Reasoning. Currently, most rule-based reasoning method research is limited to static knowledge graphs, with only a few works such as Tlogic [119] proposed by Liu et al. studying how to use rules for reasoning in temporal knowledge graphs. Temporal knowledge graph reasoning needs to mine time-dependent rules or dynamic rules, so 可以尝试将 RNN, LSTM and other models capable of modeling temporal information with existing rule learning methods to learn rules for temporal knowledge graphs. Multi-modal knowledge graphs contain multi-modal information such as text descriptions, temporal features, and image information, making rule composition more complex and representation and learning more difficult. Therefore, the next step is to research using existing multi-modal learning models to achieve automatic rule mining for multi-modal knowledge graphs through data fusion and unified representation.

Few-Shot and Zero-Shot Knowledge Graph Rule Discovery. In few-shot, one-shot, and zero-shot scenarios, rule learning is more difficult, especially high-order rules are more difficult to learn, which has a greater impact on reasoning accuracy. There are two main solutions in the future: one is to enhance the representation ability of few-shot knowledge by fusing additional information to learn higher-quality rules; the other is to use existing high-quality knowledge graphs as priors and use their high-quality samples to assist rule learning and reasoning.

Integrating Rules into Large Language Models. When LLMs are used in knowledge reasoning fields such as intelligent question answering and recommendation systems, they are easily affected by information conflicts and confusion, leading to unstable reliability in long-tail scenarios [120] and risks of factual deception. The fundamental reason is their "unconscious" use of knowledge in tasks, lacking correct and reasonable guidance. Moreover, in domain knowledge question answering and reasoning, LLMs also face the problem of "knowledge scarcity" caused by data sparsity. Therefore, how to integrate knowledge graphs and rules into LLMs to enhance pre-trained language models before, during, and after training is also a promising and potential direction for future research.

References

[1] Li Zhifei, Zhao Yue, Zhang Yan. Survey of Knowledge Graph Reasoning Based on Representation Learning. Computer Science, 2023, 50(3): 94-113.

[2] George A. Miller. WordNet: A Lexical Database for English. Communications of the ACM, 1995, 38(11): 39-41.

[3] Lehann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, Hellmann S, Morsey M, van Kleef Patrick, Aues S, Bizer C. DBPedia-A large-scale,

multiligual knowledge base extracted from wikipedia. Semantic Web, 2015, 6(2): 167-195.

[4] Fabian MS, Gjergji K, Gerhard Y. Yago: a core of semantic knowledge unifying wordnet and wikipedia. Proceedings of the 16th International World Wide Web Conference. Banff Alberta, Canada, 2007: 697-706.

[5] Mitchell T, Cohen W, Hruschka E, et al. Never-Ending Learning. Communications of the ACM, 2018, 61(5): 103-115.

[6] Wu Wentao, Li Hongsong, Wang Haixun, et al. Probase: a probabilistic taxonomy for text understanding. Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. Scottsdale Arizona, USA, 2012: 481-492.

[7] Xu Bo, Xu Yong, Liang Jiaqing, et al. CN-DBpedia: A Never-Ending Chinese Knowledge Extraction System. Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Arras, France, 2017: 428-438.

[8] Niu Xing, Sun Xinruo, Wang Haofeng, et al. Zhishi.me - Weaving Chinese Linking Open Data. Proceedings of the 10th International Semantic Web Conference. Arras, France, 2011: 205-220.

[9] Wang Xiang, He Xiangnan, Cao Yixin, et al. KGAT: Knowledge Graph Attention Network for Recommendation. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Anchorage AK, USA, 2019: 950-958.

[10] He Sun, Zou Lei, Yu Xu Jeffrey, et al. Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs. IEEE Transactions on Knowledge and Data Engineering, 2018, 30(5): 824-837.

[11] Zhou Hao, Young Tom, Huang Minlie, et al. Commonsense Knowledge Aware Conversation Generation with Graph Attention. Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 2018: 4623-4629.

[12] Liang Ke, Meng Lingyuan, Liu Meng, et al. Reasoning over Different Types of Knowledge Graphs: Static, Temporal and Multi-Modal. arXiv:2212.05767v1.

[13] Guan Saiping, Jin Xiaolong, Jia Yantao, et al. Knowledge Reasoning Over Knowledge Graph: A Survey. Journal of Software, 2018, 29(10): 2966-2994.

[14] Costas Mavromatis, George Karypis. ReaRev: Adaptive Reasoning for Question Answering over Knowledge Graphs. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, The United Arab Emirates, 2022: 2447-2458.

[15] Xian Yikun, Fu Zuohui, Muthukrishnan S, et al. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. arXiv:1906.05237.

[16] Tuan Yi-Lin, Sajjad Beygi, Maryam Fazel-Zarandi, et al. Towards Large-Scale Interpretable Knowledge Graph Reasoning for Dialogue Systems. arXiv:2203.10610.

[17] Mohamad Yaser Jaradeh, Kuldeep Singh, Markus Stocker, et al. Information extraction pipelines for knowledge graphs. Knowledge and Information Systems, 2023, 65: 1989-2016.

[18] Kenneth Marino, Ruslan Salakhutdinov, Abhinav Gupta. The More You Know: Using Knowledge Graphs for Image Classification. Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, USA, 2017: 20-28.

[19] Ma Yufeng, Xiang Nan, Dou Yajie, et al. Applications and research of knowledge graph in military system engineering. Systems Engineering and Electronics, 2022, 44(1): 146-153.

[20] Yuan Jun, Liu Guozhu, Liang Hongtao, Luo Qingcai. Summary of Research and Application of Knowledge Graphs in Risk Management Field of Commercial Banks. Computer Engineering and Applications, 2022, 58(19): 37-52.

[21] Huang Keping, Jiang Changjun. Analyzing and Reasoning Knowledge of Urban Transportation: Based on Ontology. Computer Science, 2007, 34(3): 192-196.

[22] Hu Hao, Liu Yuling, Zhang Yuchen, et al. Survey of attack graph based network security metric. Chinese Journal of Network and Information Security, 2018, 4(9): 1-16.

[23] Xia Yi, Lan Mingjing, Chen Xiaohui, et al. Survey on explainable knowledge graph reasoning methods. Chinese Journal of Network and Information Security, 2022, 8(5): 1-25.

[24] Maximilian Nickel, Kevin Murphy, Volker Tresp, et al. A Review of Relational Machine Learning for Knowledge Graphs. Proceedings of the IEEE, 2016: 11-33.

[25] Wang Q, Mao Z, Wang B, et al. Knowledge Graph Embedding: A Survey of Approaches and Applications. IEEE Transactions on Knowledge and Data Engineering, 2017, 29(12): 2724-2743.

[26] Hou Zhongni, Jin Xiaolong, Chen Jianyun, et al. Survey of Interpretable Reasoning on Knowledge Graph. Journal of Software, 2022, 33(12): 4644-4667.

[27] Ma Ang, Yu Yanhua, Yang Shengli, et al. Survey of Knowledge Graph Based on Reinforcement Learning. Journal of Computer Research and Development, 2022, 59(08): 1694-1722.

[28] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436-444.

[29] Wei Jason, Wang Xuezhi, Dale Schuurmans, et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. Proceedings of the 36th Conference on Neural Information Processing Systems, New Orleans, USA, 2022: 1-14.

[30] Bian Ning, Han Xianpei, Sun Le, et al. ChatGPT is a Knowledgeable but Inexperienced Solver: An Investigation of Commonsense Problem in Large Language Models. arXiv: 2303.16421v1.

[31] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, et al. PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311v5.

[32] Zhao Wayne Xin, Zhou Kun, Li Junyi. A Survey of Large Language Models. arXiv:2303.18223v9.

[33] Ma Ruixin, Li Zeyang, Chen Zhikui, et al. Review of Reasoning on Knowledge Graph. Computer Science, 2022, 49(S1): 74-85.

[34] Zhang Zhongwei, Cao Lei, Chen Xiliang, et al. Survey of Knowledge Reasoning Based on Neural Network. Computer Engineering and Applications, 2019, 55(12): 8-36.

[35] Sun Shuifa, Li Xiaolong, Li Weisheng, et al. Review of Graph Neural Networks Applied to Knowledge Graph Reasoning. Journal of Frontiers of Computer Science and Technology, 2023, 17(01): 27-52.

[36] Li Zhifei, Zhao Yue, Zhang Yan. Survey of Knowledge Graph Reasoning Based on Representation Learning. Computer Science, 2023, 50(3): 94-113.

[37] Song Haonan, Zhao Gang, Sun Ruoying. Developments of Knowledge Reasoning Based on Deep Reinforcement Learning. Computer Engineering and Applications, 2022, 58(1): 12-25.

[38] Li Weizhuo, Qi Guilin, Ji Qiu. Hybrid reasoning in knowledge graphs: Combining symbolic reasoning and statistical reasoning. Semantic Web, 2020, 11(1): 53-62.

[39] Zhang Wen, Chen Jiaoyan, Li Juan, et al. Knowledge Graph Reasoning with Logics and Embeddings: Survey and Perspective. arXiv:2202.07412v1.

[40] Zhang Jing, Chen Bo, Zhang Lingxi, et al. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. AI Open, 2021, 2: 14-35.

[41] Tian Ling, Zhou Xue, Wu Yanping, et al. Knowledge graph and knowledge reasoning: A systematic review. Journal of Electronic Science and Technology, 2022, 20(2): 100159.

[42] Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, et al. Learning First-Order Horn Clauses from Web Text. Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, Massachusetts, USA, 2010: 1088-1098.

[43] Salmon W, Jeffrey R, Greeno J. Statistical explanation & statistical relevance. 1. Commonwealth of Pennsylvania America: University of Pittsburgh Press, 1971.

[44] Niels Landwehr, Kristian Kersting, Luc De Raedt. Integrating Naïve Bayes and FOIL. Journal of Machine Learning Research, 2007, 8: 481-507.

[45] Niels Landwehr, Andrea Passerini, Luc De Raedt, et al. Fast learning of relational kernels. Machine Learning, 2010, 78: 305-342.

[46] Zeng Qiang, Jignesh M Patel, David Page. QuickFOIL: Scalable Inductive Logic Programming. Proceedings of the 2014 International Conference on Very Large Data Bases Endowment, Hangzhou, China, 2014: 197-208.

[47] Galárraga Luis, Teflioudi Christina, Hose Katja, et al. AMIE: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases. Proceedings of the 2013 International World Wide Web Conference Committee, Rio de Janeiro, Brazil, 2013: 413-422.

[48] Galárraga L, Telioudi C, HOSE K, et al. Fast rule mining in ontological knowledge bases with AMIE+. Proceedings of the International Journal on Very Large Data Bases, Hawaii, USA, 2015, 24(6): 707-730.

[49] Lajus J, Galárraga L, Suchanek F. Fast and exact rule mining with amie 3. Proceedings of the European Semantic Web Conference, Hersonissos, Greece, 2020: 36-52.

[50] Wang Zhichun, LI Juanzi. RDF2Rules: Learning Rules from RDF Knowledge Bases by Mining Frequent Predicate Cycles. arXiv:1512.07734.

[51] Meilicke C, Fink M, Wang Yanjie, et al. Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion. International semantic web conference. Springer, Cham, 2018: 3-20.

[52] Meilicke C, Chekol M W, Ruffinelli D, et al. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. Proceedings of the International Joint Conferences on Artifical Intelligence, Macao, China, 2019: 3137-3143.

[53] Christian Meilicke, Melisachew Wudage Chekol, Manuel Fink, Heiner Stuckenschmidt. Reinforced Anytime Bottom Up Rule Learning for Knowledge Graph Completion. arXiv:2004.04412.

[54] Ott S, Meilicke C, Samwald M. SAFRAN: An interpretable, rule-based link prediction method outperforming embedding models. arXiv:2109.08002.

[55] Sebastien Ferre. Link Prediction in Knowledge Graphs with Concepts of Nearest Neighbours. Proceedings of the European Semantic Web Conference. Portoroz, Slovenia, 2019: 84-100.

[56] Kok S, Domingos P. Learning the structure of Markov logic networks. Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 2005: 441-448.

[57] De-Raedt L, Kimmig A, Toivonen H. A probabilistic prolog and its application in link discovery. Proceedings of the 20th International Joint Conference on Artifical Intelligence, San Francisco, USA, 2007: 2462-2467.

[58] Abdus Salam, Rolf Schwitter, Mehmet A Orgun. Probabilistic Rule Learning Systems: A Survey. ACM Computing Surveys, 2021, 54(4): 1-16.

[59] Hao Jiao, Hui Xiaojing, Ma Shuo, Jin Minghui. Study on Axiomatic Truth Degree in First Order Logic. Computer Science, 2021, 48(S2): 669-671+712.

[60] Cussens J. Parameter estimation in stochastic logic program. Machine Learn, 2001, 44(3): 245-271.

[61] Wang W, Mazaitis K, Cohen W. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, San Francisco California, USA, 2013: 2129-2138.

[62] Cali A, Gottlob G, Lukasiewicz, T. A general Datalog-based framework for tractable query answering over ontologies. Journal of Web Semantics, 2012, 14: 57-83.

[63] Matthias Lanzinger, Stefano Sferrazza, Georg Gottlob, MV-Datalog+-: Effective Rule-based Reasoning with Uncertain Observations. arXiv:2202.01718.

[64] Kimmig A, Bach S H, BroechelerR M, et al. A short introduction to probabilistic soft logic. Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications, 2012: 1-4.

[65] Martin Gavalec, Zuzana Němcová, Sergeĭ Sergeev. Tropical linear algebra with the Łukasiewicz T-norm. Fuzzy Sets and Systems, 2014, 276: 131-148.

[66] Pujara J, Miao H, Getoor L, et al. Ontology-aware partitioning for knowledge graph identification. Proceedings the Workshop on Automated Knowledge Base Construction, San Francisco California, USA, 2013: 19-24.

[67] Bach S H, Broecheler M, Huang B, et al. Hinge-loss markov random fields and probabilistic soft logic. Computer Science, 2017, 18(1): 1-67.

[68] Guo Shu, Wang Quan, Wang Lihong, et al. Jointly Embedding Knowledge Graphs and Logical Rules. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin Texas, USA, 2016: 192-202.

[69] Bordes A, Usunier N, Garciaduran A, et al. Translating embeddings for modeling multi-relational data. Proceedings of the 26th International Conference on Neural Information Processing Systems, New York, USA, 2013: 2787-2795.

[70] Wang William, Cohen William. Learning First-Order Logic Embeddings via Matrix Factorization. Proceedings of the 25th International Joint Conference on Artificial Intelligence, New York, USA, 2016: 2132-2138.

[71] Guo S, Wang Q, Wang L H, et al. Knowledge graph embedding with iterative guidance from soft rules. Proceedings of the 32th AAAI Conference on Artificial

Intelligence, 2018: 4816-4823.

[72] Trouillon T, Welbil J, Riedel S, et al. Complex embeddings for simple link prediction. Proceedings of the 33rd International Conference on Machine Learning, California, USA, 2016: 2071-2080.

[73] Wang P W, Dou D J, Wu F Z, et al. Logic rules powered knowledge graph embedding. arXiv: 1903.03772.

[74] Wang Mengya, Rong Erhu, Zhuo Hankui, et al. Embedding Knowledge Graphs Based on Transitivity and Asymmetry of Rules. Proceedings of the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining, Melbourne, Australia, 2018: 141-153.

[75] Ding Boyang, Wang Quan, Wang Bin, et al. Improving Knowledge Graph Embedding Using Simple Constraints. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 2018: 110-121.

[76] Niu Guanglin, Zhang Yongfei, Li Bo, et al. Rule-Guided Compositional Representation Learning on Knowledge Graphs. Proceedings of the AAAI Conference on Artificial Intelligence, 2020, 34(03): 2950-2958.

[77] Tang Xiaojuan, Zhu Song-Chun, Liang Yitao, et al. RulE: Neural-Symbolic Knowledge Graph Reasoning With Rule Embedding. arXiv:2210.14905v1.

[78] Zhang Wen, Bibek Paudel, Wang Liang, et al. Iteratively Learning Embeddings and Rules for Knowledge Graph Reasoning. Proceedings of the World Wide Web Conference, New York, USA, 2019: 2366-2377.

[79] Ho V T, Stepanova D, Gad-Elrab M H, et al. Rule Learning from Knowledge Graphs Guided by Embedding Models. Proceedings of International Semantic Web Conference, Monterey, USA, 2018: 72-90.

[80] Omran P G, Wang K, Wang Z. An Embedding-Based Approach to Rule Learning in Knowledge Graphs. IEEE Transactions on Knowledge and Data Engineering. 2021, 33(4): 1348-1359.

[81] Cheng Kewei, Yang Ziqing, Zhang Ming, et al. UniKER: A Unified Framework for Combining Embedding and Definite Horn Rule Reasoning for Knowledge Graph Inference. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 2021: 9753-9771.

[82] Wang Peifeng, Han Jialong, Li Chenliang, et al. Logic Attention Based Neighborhood Aggregation for Inductive Knowledge Graph Embedding. Proceedings of the 19th AAAI Conference on Artificial Intelligence, Hawaii, USA, 2019: 7152-7159.

[83] Sun Haitian, Bhuwan Dhingra, Manzil Zaheer, et al. Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text. Proceedings of

the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 2018: 4231-4242.

[84] Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Computation, 1997, 9(8): 1735-1780.

[85] Wang Hanzhi, Wei Zhewei, Gan Junhao, et al. Personalized PageRank to a Target Node. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, USA, 2020: 657-667.

[86] He Yongquan, Wang Zihan, Zhang Peng, et al. VN Network: Embedding Newly Emerging Entities with Virtual Neighbors. Proceedings of the International Conference on Information and Knowledge Management, Virtual Event, Ireland, 2020: 505-514.

[87] Zhao Yunxiang, Qi Jianzhong, Liu Qingwei, et al. WGCN: Graph Convolutional Networks with Weighted Structural Features. Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, 2021: 624-633.

[88] Cho Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 2014: 1724-1734.

[89] Qu Meng, Tang Jian. Probabilistic Logic Neural Networks for Reasoning. Proceedings of the 33rd Conference on Neural Information Processing Systems, Vancouver, Canada, 2019: 7712-7722.

[90] Zhang C, Bütepage J, Kjellström H. Advances in Variational Inference. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2019: 41(8): 2008-2026.

[91] Veličković P, Casanova A, Liò P. Graph attention networks. Proceedings of the 6th International Conference on Learning Representations, Vancouver CANADA, 2018: 1-12.

[92] Vardhan Harsha, Vivek L, Guo Jia, et al. Probabilistic Logic Graph Attention Networks for Reasoning. Proceedings of the International World Wide Web Conference Committee, Taipei, Taiwan, 2020: 669-673.

[93] Zhang Yuyu, Chen Xinshi, Yang Yuan, et al. Efficient Probabilistic Logic Reasoning with Graph Neural Networks. Proceedings of the International Conference on Learning Representations, Kigali, Rwanda, 2023: 1-20.

[94] Lei Deren, Jiang Gangrong, Gu Xiaotao, et al. Learning Collaborative Agents with Rule Guidance for Knowledge Graph Reasoning. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, Online, 2020: 8541-8547.

[95] Xia Yi, Lan Mingjing, Luo Junyong. Iterative rule-guided reasoning over sparse knowledge graphs with deep reinforcement learning. Information Processing and Management, 2022, 59: 103040.

[96] Gallaire Herve, Minker Jack, Nicolas Jean-Marie. Logic and Databases: A Deductive Approach. ACM Computing Surveys. 1984, 16(2): 153-185.

[97] Tim Rocktäschel, Sebastian Riedel. End-to-End Differentiable Proving. Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, USA, 2017: 3791-3803.

[98] Pasquale Minervini, Matko Bosnjak, Tim Rocktaschel, et al. Differentiable Reasoning on Large Knowledge Bases and Natural Language. Proceedings of the AAAI Conference on Artificial Intelligence, New York, USA, 2020: 5182-5190.

[99] Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, et al. Learning reasoning strategies in end-to-end differentiable proving. Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, 2020: 6938-6949.

[100] Cohen W W. Tensorlog: a differentiable deductive database. arXiv: 1605.06523.

[101] Yang Fan, Yang Zhilin, Cohen William W. Differentiable learning of logical rules for knowledge base reasoning. Proceedings of the Advances in Neural Information Processing Systems, LA, USA, 2017: 2319-2328.

[102] Wang Po-Wei, Daria Stepanova, Csaba Domokos. Differentiable learning of numerical rules in knowledge graphs. Proceedings of the 8th International Conference on Learning Representations, Virtual, 2020: 1-12.

[103] Ali Sadeghian, Mohammadreza Armandpour, Ding Patrick, et al. DRUM: End-To-End Differentiable Rule Mining On Knowledge Graphs. Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, Canada, 2019: 15347-15357.

[104] Yang Yuan, Song Le. Learn to explain efficient via neural logic inductive learning. Proceedings of the 8th International Conference on Learning Representations, Virtual, 2020: 1-15.

[105] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention Is All You Need. Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, USA, 2017: 6000-6010.

[106] Dong Honghua, Mao Jiayuan, Lin Tian, et al. Neural Logic Machines. Proceedings of the 7th International Conference on Learning Representations, Virtual, 2019: 1-22.

[107] Chen Hanxiong, Li Yunqi, Shi Shaoyun, et al. Graph Collaborative Reasoning. Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, Canada, 2019: 75-84.

[108] Teru Komal K, Denis Etienne G, Hamilton William L. Inductive relation prediction by subgraph reasoning. Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, 2020: 9448-9457.

[109] Mai Sijie, Zheng Shuangjia, Yang Yuedong, et al. Communicative Message Passing for Inductive Relation Reasoning. Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, Canada, 2021: 4294-4302.

[110] Jiajun Chen, Huarui He, Feng Wu, et al. Topology-Aware Correlations Between Relations for Inductive Link Prediction in Knowledge Graphs. Proceedings of the 35th AAAI Conference on Artificial Intelligence, Vancouver, Canada, 2021: 6271-6278.

[111] Lin Qika, Liu Jun, Xu Fangzhi, et al. ConGLR: Incorporating Context Graph with Logical Reasoning for Inductive Relation Prediction. Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, 2022: 893-903.

[112] Xu Xiaohan, Zhang Peng, He Yongquan, et al. Subgraph Neighboring Relations Infomax for Inductive Link Prediction on Knowledge Graphs. Proceedings of the 31st International Joint Conference on Artificial Intelligence Main Track, Shenzhen, China, 2022: 2341-2347.

[113] Zha Hanwen, Chen Zhiyu, Yan Xifeng, et al. Inductive Relation Prediction by BERT. Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2022: 5923-5931.

[114] Pan Yudai, Liu Jun, Zhang Lingling, et al. Learning First-Order Rules with Relational Path Contrast for Inductive Relation Reasoning. arXiv:2110.08810.

[115] Yan Zuoyu, Ma Tengfei, Gao Liangcai, et al. Cycle Representation Learning for Inductive Relation Prediction. Proceedings of the International Conference on Learning Representations, Virtual, 2022: 1-16.

[116] Qu Meng, Chen Junkun, Louis-Pascal Xhonneux, et al. RNNLogic: learning logic rules for reasoning on knowledge graphs. Proceedings of the 2021 International Conference on Learning Representations, Virtual, 2021.

[117] Huang Qian, Ren Hongyu, Jure Leskovec. Few-shot Relational Reasoning via Connection Subgraph Pretraining. Proceedings of the 36th Conference on Neural Information Processing Systems, New Orleans, LA, USA, 2022: 6397-6409.

[118] Du Zhengxiao, Zhou Chang, Yao Jiangchao, et al. CogKR: Cognitive Graph for Multi-Hop Knowledge Reasoning. IEEE Transactions on Knowledge and Data Engineering. 2021, 35(2): 1283-1295.

[119] Liu Y, Ma Y, Hildebrandt M, et al. TLogic: Temporal Logical Rules for Explainable Link Forecasting on Temporal Knowledge Graphs. Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2022: 4120-4127.

[120] Bang Yejin, Samuel Cahyawijaya, Nayeon Lee, et al. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. arXiv:2302.04023.

[121] Toutanova K, Chen D. Observed versus latent features for knowledge base and text inference. Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, Beijing, China, 2015: 57-66.

[122] Dettmers T, Minervini P, Stenetorp P, et al. Convolutional Knowledge Graph Embeddings. Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, USA, 2018: 1811-1818.

[123] Xiong W, Hoang T, Wang W Y. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 2017: 564-573.

[124] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning. 1992, 8(3–4): 229-256.

[125] Nickel Maximilian, Tresp Volker, Kriegel Hans-Peter. A Three-Way Model for Collective Learning on Multi-Relational Data. Proceedings of the 28th International Conference on International Conference on Machine Learning, Bellevue Washington, USA, 2011: 809-816.

[126] Schlichtkrull M, Kipf T N, Bloem P, et al. Modeling Relational Data with Graph Convolutional Networks. Proceedings of the 15th European Semantic Web Conference, Heraklion, Greece, 2018: 10843.

(Corresponding author: Cheng Qing, E-mail: zzfzad2021@163.com)

Author Contribution Statement: Zeng Zefan: Literature research and analysis, paper drafting, final version revision; Si Yuehang: Literature research; Liu Zhong: Paper writing guidance; Cheng Qing: Research proposition and design.

Note: Figure translations are in progress. See original paper for figures.