KNOWLEDGE GRAPH REASONING: ALGORITHMS AND APPLICATIONS

SHENGYUAN CHEN

PhD

The Hong Kong Polytechnic University

2024

The Hong Kong Polytechnic University

Department of Computing

Knowledge Graph Reasoning: Algorithms and Applications

Shengyuan Chen

A thesis submitted in partial fulfillment of the requirements for

the degree of Doctor of Philosophy

July 2024

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Shengyuan Chen

Abstract

Knowledge graphs (KGs) serve as powerful tools in various domains by capturing and organizing complex relationships between entities. Yet, despite their utility, KGs often remain incomplete, limiting their full potential. KG completion and KG alignment emerge as critical solutions to fill these gaps, and at the heart of both lies KG reasoning. Traditional reasoning methods, whether neural or symbolic, often falter when tasked with large-scale KGs due to challenges in scalability and interpretability. Hence, advancing the efficiency and effectiveness of KG reasoning algorithms is paramount for unlocking their broader applications.

This thesis presents several groundbreaking approaches designed to enhance KG reasoning. First, we introduce DiffLogic, a differentiable neuro-symbolic reasoning framework that fuses the strengths of symbolic rule-based inference with the flexibility of neural networks. By modeling the joint probability of truth scores through weighted logic rules embedded within a Markov random field, DiffLogic achieves efficient rule grounding, allowing it to perform scalable inference over expansive KGs with precision. Building upon this, we propose NeuSymEA, a robust entity alignment method that bridges the gap between symbolic and neural reasoning through variational inference. NeuSymEA is uniquely equipped to operate in low-resource environments, aligning entities even when limited labeled data is available. Through logic decomposition and deduction, NeuSymEA scales symbolic reasoning to handle long rule inferences, all while ensuring interpretability and improving precision in alignment tasks. Lastly,

we introduce LLM4EA, a label-free entity alignment framework that harnesses the power of large language models (LLMs) to address noisy annotations. By actively selecting key entities for annotation and refining label accuracy through probabilistic reasoning, LLM4EA enhances alignment robustness, minimizing the effect of noise while maximizing performance.

The research presented in this thesis pushes the boundaries of what's possible in KG reasoning, offering scalable and efficient solutions for constructing and exploiting large KGs. The advancements presented in this work open new doors for robust and interpretable applications across fields such as information retrieval, social computing, health informatics, and bioinformatics, enabling the extraction of deeper insights from the vast and interconnected data that knowledge graphs offer.

Publications Arising from the Thesis

1. Shengyuan Chen, Yunfeng Cai, Huang Fang, Xiao Huang, Mingming Sun, "Differentiable Neuro-Symbolic Reasoning on Large-Scale Knowledge Graphs", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

2. Shengyuan Chen, Qinggang Zhang, Junnan Dong, Wen Hua, Qing Li, Xiao Huang, "Entity Alignment with Noisy Annotations from Large Language Models", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

3. Shengyuan Chen, Qinggang Zhang, Junnan Dong, Wen Hua, Jiannong Cao, Xiao Huang, "Neuro-symbolic Entity Alignment via Variational Inference", manuscript submitted to *The International Conference on Learning Representations (ICLR)*.

Acknowledgments

I would like to express my deepest gratitude to everyone who has supported me throughout the journey of my PhD.

First and foremost, I would like to thank my advisor, Dr. Xiao Huang, for his unwavering support, guidance, and encouragement. His insightful feedback and constructive criticism have been invaluable in shaping this research. I am also grateful for the freedom he provided me to explore my ideas, which significantly contributed to my personal and professional growth. I am also thankful to my co-supervisor, Dr. Qing Li, for his support and assistance during my PhD journey.

I extend my sincere thanks to the members of my dissertation committee, Dr. Wen Hua, Dr. Yangqiu Song, and Dr. Di Jin, for their time, expertise, and valuable suggestions. Their diverse perspectives have greatly enhanced the quality of my work.

I am incredibly thankful to my labmates at the DEEP Lab for their camaraderie and constant help. Their valuable advice has been extremely helpful for my work.

A heartfelt thanks goes to my family for their endless love, patience, and encouragement. They have been my anchor through the highs and lows of this journey.

Finally, I dedicate this thesis to everyone who believed in me and supported me in various ways. This achievement would not have been possible without you all.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Knowledge graph (KG) reasoning refers to the process of using existing facts in a KG to infer new knowledge that is not explicitly stated in the original KG [40]. For instance, if we know that *(Paris, is_the_capital_of, France)* and *(France, is_a_country_in, Europe)*, we can infer that *(Paris, is_a_city_in, Europe)*, which is not explicitly included in the original KG. KG reasoning is essential to KG completion [43] and KG error detection [62, 13]. Moreover, KG reasoning can infer underlying knowledge and improve the quality of learning or predictions on KGs, which benefits various downstream tasks, such as question answering [22, 12], recommendations [21, 58], and interpretable machine learning [26, 51].

From the perspective of KG construction, KG reasoning is crucial in developing larger and more comprehensive knowledge graphs. By leveraging its ability to infer new facts from existing ones, KG reasoning facilitates fundamental tasks such as KG completion and KG alignment. This reasoning enriches the graph with inferential connections and insights, enhancing the overall quality and applicability of the knowledge graph. Additionally, KG reasoning can handle uncertainties within the KG and eliminate

erroneous noise in the data, thereby strengthening the stability of KG completion and KG alignment processes in constructing larger KGs.

Specifically, KG completion leverages KG reasoning to predict and fill in missing information, making the knowledge graph more comprehensive and useful. This process is vital for ensuring the KG's accuracy and completeness, which are essential for its practical applications. In contrast, KG alignment uses KG reasoning to integrate disparate knowledge graphs by aligning entities and relationships across them, creating a unified and cohesive knowledge structure. This alignment process is critical for consolidating information from multiple sources into a single, comprehensive KG.

Through these processes, KG reasoning not only enhances the current state of KGs but also lays a robust foundation for advanced applications in areas such as question answering, recommendations, and interpretable machine learning. By improving the completeness, accuracy, and integration of knowledge graphs, KG reasoning significantly contributes to the development of larger and more comprehensive knowledge systems. Consequently, KG reasoning plays a pivotal role in constructing and refining KGs, thereby enabling more sophisticated and reliable applications that rely on these knowledge structures.

1.2 Knowledge Graph Completion at Scale: Differentiable Neuro-symbolic Reasoning on Knowledge Graphs

There are mainly two lines of research in KG reasoning [65, 64]. First, rule-based reasoning derives new triples from existing ones by applying a set of predefined rules, which are usually expressed in a form of logical statements [59, 39, 16]. For example, if there are two triples *(A, is_a_parent_of, B)* and *(B, is_a_parent_of, C)*, and a

rule saying "the parent of a parent is a grandparent", then we can infer a new triple *(A, is_a_grandparent_of, C)*. Rule-based reasoning is particularly effective when the rules are well-defined and the KG is relatively small and static [1]. Recent studies employ Markov logic network (MLN) [38] to dynamically learn soft rules, which are associated with weight scores indicating their credibility. But it remains not scalable because assessing the truth scores to all possible triples requires $\mathcal{O}(|\mathcal{E}|^2|\mathcal{R}|)$ parameters, where $|\mathcal{E}|$ and $|\mathcal{R}|$ denote the numbers of entities and relations. Thus, rule-based reasoning becomes inefficient when dealing with large KGs. Second, KG-embedding based reasoning projects a KG into a low-dimensional space and infers new relations based on embedding representations [67, 37]. It assumes that KG embedding can preserve most semantic information in the original KGs [3, 54, 49, 15] so that missing relations can be inferred by using the distances or semantic matching scores in the embedding space. Recent studies [41, 52, 63] also seek to design tailored Graph convolutional networks for learning structural-aware KG embeddings. KG-embedding based reasoning can be scalable [70, 37]. However, there is no explicit rule to ensure its accuracy and logical consistency. Also, it requires a sufficient amount of training data, and its results are difficult to interpret.

Neuro-symbolic reasoning models combine the advantages of both rule-based and embedding-based techniques, which is to approximate the predicted truth scores of all possible triples inferred by rules with the normalized output scores of a KG-embedding model. However, it is nontrivial to effectively and efficiently construct such an approximation. Directly employing rules [17, 18] to regularize the embedding learning is efficient, but it cannot update rule weights and is thus sensitive to the initialization of rule weights. Incorporating KG-embedding models into sophisticated rule-based models, such as MLN, enables the handling of rule uncertainty [36, 66]. However, directly approximating the distributions of the truth scores are still challenging. First, directly approximating the distribution of truth scores within the MLN framework is unfeasible. This process necessitates the optimization of the joint probability of the

approximated distribution, which subsequently requires the computation of integration across all scores. Second, optimizing neuro-symbolic models rely on training with ground formulas (i.e. instantiated rules). Given the large grounding space, the performance of neuro-symbolic models may deteriorate if important ground formulas are missed [17, 36], and efficiency issues may arise if too many ground formulas are considered [66].

To this end, we proposed a differentiable framework - DiffLogic [42]. Firstly, a tailored filter is used to adaptively select important ground formulas based on weighted rules and extract connected triples. Second, a KG-embedding model is used to compute a truth score for each triple. Then we employ a continuous MLN named probabilistic soft logic (PSL) [1] that takes these truth scores as input, and assesses the overall agreement among rules, weights, and observed triples with a joint probability. The PSL template enables end-to-end differentiable optimization. In this way, DiffLogic is optimized by alternately updating embedding and weighted rules. We present the detailed problem statement and methodology in Chapter 3.

1.3 Robust Entity Alignment: Neuro-symbolic Entity Alignment via Variational Inference

As another way to address the sparsity and incompleteness issues in KGs, entity alignment (EA) aims to merge disparate KGs into a unified, comprehensive knowledge base by identifying and linking equivalent entities across different KGs. For instance, by aligning entities between a financial KG and a legal KG, EA facilitates the understanding of complex relationships, such as identifying the same corporations across the two KGs to assess how legal regulations impact their financial performance. This alignment enables a more nuanced exploration and interrogation of interconnected data, providing richer insights into how entities operate across multiple domains.

To perform entity alignment, existing methods can be broadly categorized into symbolic models and neural models. Symbolic models [44, 24, 34] provide interpretable and precise inference by mining ground rules, but they struggle with aligning long-tail entities, especially those without aligned neighbors. In such cases, the lack of supporting rules leads to low recall. Conversely, neural models, such as translation models [5, 47] and Graph Convolutional Networks (GCNs) [33, 31, 55, 32, 29, 28], excel in recalling similar entities by embedding them in a continuous space, yet they often fail to distinguish entities with similar representations, causing a drop in precision as the entity pool grows. Neuro-symbolic models aim to combine the strengths of both approaches, offering the interpretability and precision of symbolic models alongside the high recall capabilities of neural models.

However, neuro-symbolic reasoning in entity alignment (EA) faces several challenges. First, combining symbolic and neural models into a unified framework is suboptimal due to the difficulty in aligning their objectives. Current approaches either use neural models as auxiliary modules for symbolic models to measure entity similarity [34] or employ symbolic models to refine pseudo-labels [30, 6]. Second, in EA task, the search space for rules is large, as the EA task requires deriving ground rules from both intra-KG and inter-KG structural patterns, leading to an exponentially large search space with increasing rule length. Finally, generating interpretations for EA remains underexplored. Effective interpretations should not only generate supporting rules but also quantify their confidence through rule weights.

To overcome these challenges, we propose NeuSymEA [**?**], a neuro-symbolic framework that combines the strengths of both symbolic and neural models. NeuSymEA models the joint probability of truth score assignment for all possible entity pairs using a Markov random field, regulated by a set of weighted rules. This joint probability is optimized via a variational EM algorithm. During the E-step, a neural model parameterizes the truth scores and infers the missing alignments. In the M-step, the rule weights are updated based on both observed and inferred alignments. To

leverage long rules without suffering from the exponential search space, we employ logic deduction to decompose rules of any length into shorter, unit-length rules. This allows for efficient inference and weight updates for long rules. After training, the learned rules are adapted into an explainer, enhancing interpretability. Specifically, we reverse the logic deduction process to calculate the weights of long rules based on the learned weights of their shorter counterparts. Additionally, an explainer is introduced to efficiently generate supporting rules with quantified confidence through breadth-first search, as explanations for inferred alignments. We present the details of this model in Chapter 4

1.4 Label-free Entity Alignment: Entity Alignment with Noisy Annotations from Large Language Models

While we developed a robust and interpretable entity alignment model, it relies on accurate seed alignments for training. Acquiring these seed alignments is still prohibitively expensive due to the need for extensive cross-domain knowledge. Thus, we propose a label-free entity alignment framework that facilitates effective learning for any off-the-shelf EA model.

Recently, Large Language Models (LLMs) have showcased their superior capability in processing semantic information, which has significantly advanced various graph learning tasks [27] such as node classification [7, 28], graph reasoning [68, 4], recommender systems [72, 56], SQL query generation [60, 20], and knowledge graph-based question answering [53, 61, 14]. Their capacity to extract meaningful insights from graph data opens up new possibilities for automating EA. Notably, recent studies [50, 71, 69, 23] have explored the use of LLMs in EA, primarily focusing on finetuning a pretrained LLM such as Bert to learn semantic-aware representations, relying on accurate seed

alignments as training labels. Yet, the potential of LLMs for label-free EA via in-context learning remains unexplored.

However, directly applying LLMs to automate EA poses significant challenges. Firstly, conventional EA models presume that all annotations are correct; yet, LLMs can generate false labels due to LLMs' inherent randomness and the potential incompleteness or ambiguity in the semantic information of entities. Training an EA model directly on these noisy labels can severely impair the final alignment performance. Secondly, given the vast number of entity pairs, annotation with LLMs would be prohibitively expensive. Maximizing the utility of a limited LLM query budget is essential. Existing solutions such as active learning cannot be directly applied since the annotations are noisy.

In response to the outlined challenges, we introduce LLM4EA [6], a unified framework designed to effectively learn from noisy pseudo-labels generated by LLMs while dynamically optimizing the utility of a constrained query budget. LLM4EA actively selects source entities based on feedback from a base EA model, focusing on those that significantly reduce uncertainty for both the entities themselves and their neighbors. This approach allocates the query budget to important entities, guided by the intra-KG and inter-KG structure. To manage the noisy pseudo-labels effectively, LLM4EA incorporates an unsupervised label refiner that enhances label accuracy by selecting a subset of confident pseudo-labels through probabilistic reasoning. These refined labels are then utilized to train the base EA model for entity alignment. The confident alignment results inferred by the EA model inform active selection in subsequent iterations, thereby progressively improving the framework's effectiveness in a coherent and integrated manner. We depict how probabilistic reasoning enhance entity alignment with large language models in Chapter 5.

1.5 Correlations among three work

1.5.1 Task-Level Correlations

The three research works—DiffLogic, NeuSymEA, and LLM4EA—are interrelated in their collective aim to improve knowledge graph (KG) quality, which ultimately facilitates downstream tasks such as recommendation systems, question answering, and drug discovery. Each approach targets different challenges that, when addressed, enhance the completeness, accuracy, and usability of KGs:

- **DiffLogic**: This work focuses on KG completion at scale by filling in missing facts within KGs. Leveraging neuro-symbolic reasoning, DiffLogic combines neural networks with rule-based logic to efficiently infer missing information and ground rules. By ensuring parameter efficiency and scalability, DiffLogic improves the quality of KGs with minimal computational overhead.
- **NeuSymEA**: Concentrating on robust entity alignment, NeuSymEA enhances the process of aligning entities across different KGs, even in scenarios with limited labeled data. It combines neural models with symbolic logic, ensuring precise alignment—a critical factor for merging multiple KGs into a unified, high-quality graph. NeuSymEA's scalability to long, complex rules via logic deduction further improves the overall structure and accuracy of KGs.
- **LLM4EA**: This work addresses label-free entity alignment by utilizing large language models (LLMs) to handle noisy and incomplete annotations. Through automated alignment processes, including active sampling and probabilistic label refinement, LLM4EA minimizes noise and dependency on extensive manual labeling. This approach significantly improves KG quality by making entity alignment more robust and scalable.

In summary, all three works share the overarching goal of improving KG quality

by addressing distinct aspects: KG completion (DiffLogic), robust entity alignment (NeuSymEA), and label-free entity alignment (LLM4EA). Together, they contribute to the enhanced utility of KGs in various downstream AI applications.

1.5.2 Methodology-Level Correlations

At the methodology level, the three works—DiffLogic, NeuSymEA, and LLM4EA—are fundamentally connected through the neuro-symbolic reasoning framework, though each applies this methodology in unique ways to tackle different aspects of KG refinement.

Neuro-Symbolic Reasoning

All three works integrate symbolic logic with neural networks, providing a methodological foundation for combining the strengths of both paradigms. Symbolic models offer logical structure and interpretability, while neural networks provide scalability and handle uncertainty and noise. This combination allows the models to operate effectively on structured knowledge (e.g., logic rules) and unstructured or noisy data (e.g., KG embeddings, noisy annotations).

Rule-Based Reasoning

- **DiffLogic**: Rules are employed to guide the completion of missing facts in KGs. Truth scores of facts are parameterized and optimized using a combination of neural networks and weighted rules within a Markov random field (MRF). By grounding and optimizing these rules in a differentiable manner, DiffLogic scales efficiently to large KG completion tasks.
- **NeuSymEA**: Rule-based reasoning is extended to entity alignment. Here, long rules are decomposed and composed via logic deduction, enabling efficient align-

ment by leveraging logical relationships between entities across KGs. NeuSymEA uses neural models to refine alignment scores, ensuring scalability to long and complex rule structures.

- **LLM4EA**: While LLM4EA doesn't directly ground rules like the other two, it still employs probabilistic reasoning for label refinement. Through active learning and continuous improvement of noisy labels provided by LLMs, it achieves accurate entity alignment in a label-free manner.

Optimization Techniques

DiffLogic and NeuSymEA both rely on differentiable optimization techniques. In DiffLogic, a hinge-loss MRF models the joint probability of truth scores, with t-norms used to relax truth scores into a continuous space, enabling efficient gradient-based optimization.

NeuSymEA uses a variational EM framework to jointly optimize the neural and symbolic components. This allows for iterative refinement of both the neural model's parameters and symbolic rule weights, enhancing alignment performance.

In LLM4EA, although the focus is on probabilistic reasoning rather than directly optimizing rules, probabilistic models are used to refine noisy annotations. This reasoning approach adjusts and improves the entity alignment process continuously.

Handling Uncertainty and Noise

DiffLogic addresses uncertainty by allowing soft reasoning through the neural parameterization of truth scores and differentiable rule grounding.

NeuSymEA handles uncertainty by using neural models to supplement weak or missing logical evidence, particularly in low-resource scenarios. It augments sparse anchor pairs with inferred pairs from the neural model, ensuring robustness.

LLM4EA directly tackles noise in annotations by utilizing LLMs to generate initial labels and then employing probabilistic label refinement to correct and improve those labels, making entity alignment more robust and accurate.

Conclusion

At the methodology level, the correlation among DiffLogic, NeuSymEA, and LLM4EA is rooted in their shared use of neuro-symbolic reasoning, combining logical rules and neural networks to enhance different tasks within KG refinement. Each work applies this methodology to a specific challenge—KG completion, entity alignment, and noisy label refinement—using techniques such as rule-based reasoning, differentiable optimization, and probabilistic refinement to handle uncertainty and scale the reasoning process effectively.

1.6 Summary of Contributions

The contributions of DiffLogic, NeuSymEA, and LLM4EA are significant in advancing the fields of knowledge graph reasoning and entity alignment. Here is a breakdown of the contributions from each work:

1.6.1 DiffLogic

- **Efficient Rule Grounding**: DiffLogic introduces a novel rule-guided iterative grounding technique that significantly reduces the number of ground rules needed during reasoning. This technique selectively identifies important ground formulas, making the reasoning process more efficient while maintaining performance.
- **Differentiable Optimization**: The framework combines neural embeddings and rule-based symbolic logic within a Markov random field (MRF), enabling

scalable, differentiable optimization over large knowledge graphs. This allows for continuous parameterization of truth scores, making the model more adaptable to large-scale applications.

- **Improved Scalability**: DiffLogic effectively balances the trade-offs between rule-based reasoning and neural embeddings, enabling it to scale to large KGs while retaining logical consistency and parameter efficiency.

1.6.2 NeuSymEA

- **Joint Neural-Symbolic Framework**: NeuSymEA combines symbolic and neural approaches for entity alignment, integrating them in a variational EM framework. This allows it to model the joint probability of entity alignments by leveraging both learned embeddings and symbolic rules.

- **Scalability to Long Rules**: The framework introduces logic deduction to decompose long rules into shorter, more manageable units. This increases the model's capacity to handle long and complex rules while maintaining scalability.

- **Robustness in Low-Resource Scenarios**: NeuSymEA demonstrates robustness in label-limited scenarios, augmenting sparse anchor pairs through inferred pairs from the neural model. This enhances alignment precision even with limited training data.

- **Interpretable Inference**: NeuSymEA provides explainability by generating supporting rules for each entity alignment with quantified confidence scores. This makes the alignment process interpretable, adding value to applications where transparency is important.

1.6.3 LLM4EA

- **Label-Free Entity Alignment**: LLM4EA addresses the challenge of label-free entity alignment by leveraging large language models (LLMs) to generate noisy annotations. This reduces the dependency on manually curated seed alignments, which are often expensive to obtain.

- **Probabilistic Label Refinement**: The model incorporates a probabilistic reasoning-based label refiner to continuously improve the quality of noisy labels generated by LLMs. This ensures that even noisy data can be used effectively for entity alignment.

- **Active Sampling for Efficient Annotation**: To optimize the limited annotation budget, LLM4EA employs active sampling techniques to focus on the most important entities, maximizing the utility of each LLM query and improving alignment performance while minimizing costs.

In summary, **DiffLogic**, **NeuSymEA**, and **LLM4EA** contribute to advancing knowledge graph quality by addressing different challenges within the neuro-symbolic reasoning framework. **DiffLogic** introduces an innovative differentiable reasoning framework that efficiently combines symbolic logic with neural embeddings, enabling scalable knowledge graph completion. It achieves this through a novel rule-guided iterative grounding technique, which significantly reduces computational overhead while maintaining high performance on large-scale knowledge graphs. Building on this foundation, **NeuSymEA** extends the neuro-symbolic reasoning paradigm to the task of entity alignment. By integrating neural models with symbolic reasoning through a variational EM framework, NeuSymEA enhances the robustness and precision of alignment, even in low-resource settings, while providing interpretability through rule-based explanations. Finally, **LLM4EA** tackles the problem of noisy annotations in entity alignment by leveraging large language models to perform label-free

alignment. With active sampling and probabilistic label refinement, it minimizes the reliance on expensive manual labeling and handles noisy data effectively, ensuring that entity alignment remains robust and scalable. Together, these three works provide a comprehensive approach to improving knowledge graph quality by addressing its core challenges: completion, alignment, and the handling of noisy data, ultimately facilitating more accurate and scalable downstream applications.

Chapter 2

Related Work

2.1 Knowledge Graph Reasoning

Neuro-symbolic reasoning for KG completion. There have been some studies attempting to integrate rule-based methods and KG-embedding models. For the KG completion task, [17] proposes to learn embeddings from both triples and rules by treating triples as 'atomic formulas' while treating ground logic formulas as 'complex formulas', thus unifying the learning from triples and rules. However, their framework only uses hard rules and thus cannot make use of the soft rules with uncertainty. Another study [18] applied soft rules for generating additional training data but could not optimize rule weights, making the model's effectiveness dependent on rule weights initialization. [36] enable embedding learning and rule weight updating in a tailored MLN framework, by alternately employing one component to annotate triples to update the other component. However, the annotation process renders the inference not differentiable and is sensitive to the annotation threshold. Moreover, their grounding process only considers ground formulas with premise atoms observed in the training set, limiting the model's effectiveness due to the potential omission of important ground formulas. [66] designed a graph neural network for learning structure-

aware expressive representations under the MLN framework. However, its inference efficiency suffers from a large number of ground formulas and limited generalization ability (requiring querying the test data during inference). The MLN-based neuro-symbolic models [36, 66] benefit from the dynamically updated rule weights which handle uncertainty. However, they all fail to directly optimize the objective of MLN due to the complexity of the associated integration and thereby resorting to optimizing ELBO. Despite extensive studies of neuro-symbolic reasoning in KG completion[11, 9], they only consider single-KG structures, thus cannot be directly adopted to entity alignment which requires consideration of inter-KG structures.

Neuro-symbolic reasoning for entity alignment. Recent models have sought to combine symbolic and neural approaches for the entity alignment task. For instance, [34] enhances probabilistic reasoning by utilizing KG embeddings, employing a KG-embedding model to measure similarities during both updating and inference processes. [30] implements self-bootstrapping with pseudo-labeling in a neural framework, using rules to choose confident pseudo-labels. However, it relies solely on unit-length rules, which restricts its effectiveness for long-tail entities. Furthermore, neither of these approaches offers interpretability for the inferred alignments.

Probabilistic Reasoning. In the literature on knowledge graph reasoning, probabilistic reasoning has been effectively applied to infer new soft labels and their associated probabilities from existing labels. It has been utilized in domains such as knowledge graph completion [36, 42, 66] and entity alignment [44, 34], where it naturally represents complex relational patterns with simple rules and performs precise inferences. In our work in LLM4EA, however, due to the potential inaccuracies in the pseudo-labels generated by LLMs, the newly inferred alignments may be incorrect. Consequently, we opt not to employ probabilistic reasoning directly for the entity alignment task. Instead, we emphasize its use primarily for filtering false pseudo-labels that demonstrate structural incompatibilities within our framework.

2.2 Entity Alignment with Large Language Models

Recent approaches have sought to leverage LLMs for entity alignment in knowledge graphs, primarily focusing on integrating structural and semantic information for improved alignment performance. BERT-INT [50] fine-tunes a pretrained BERT model to capture interactions and attribute information between entities. Similarly, SDEA [71] employs a pretrained BERT to encode attribute data, while integrating neighbor information via a GRU to enhance structural representation. TEA [69] reconceptualizes entity alignment as a bidirectional textual entailment task, utilizing pretrained language models to estimate entailment probabilities between unified textual sequences representing entities. A novel approach, ChatEA [23], introduces a KG-code translation module that converts knowledge graph structures into a format comprehensible to LLMs, enabling these models to apply their extensive background knowledge to boost the accuracy of entity alignment. Notably, these models primarily focus on fine-tuning pretrained language models using a set of training labels and do not exploit the zero-shot capabilities of LLMs. In contrast, our proposed model, LLM4EA, leverages the zero-shot potential of LLMs, enabling it to generalize to new datasets without the need for labeled data.

Chapter 3

Differentiable Neuro-Symbolic Reasoning on Large-Scale Knowledge Graphs

In this chapter, we formally define the problem of knowledge base completion, provide a brief introduction to first-order logic, and how to evaluate its agreement with a knowledge base. Then propose a differentiable neuro-symbolic reasoning model – DiffLogic, to tackle this problem.

3.1 Problem Statement

A knowledge base $\mathcal{K}$ comprises a set of entities $\mathcal{E}$ and a set of relations $\mathcal{R}$. For any pair of head-tail entities $(h, t) \in \mathcal{E} \times \mathcal{E}$ and a relation $r \in \mathcal{R}$, the relation maps the pair of entities to a binary score, i.e., $r : \mathcal{E} \times \mathcal{E} \to \{0, 1\}, \forall r \in \mathcal{R}$, indicating that the head entity h either has the relation r with the tail entity t or not. In the knowledge base completion problem, people observe a set of facts $\mathcal{O} = \{(h_i, r_i, t_i)\}_{i=1}^{n}$ along with their true assignment vector $\boldsymbol{x} = [x_1, \ldots, x_{|\mathcal{O}|}] \in \mathbb{R}^{|\mathcal{O}|}$ with $x_i = r_i(h_i, t_i)$. Denote the

unobserved facts as $\mathcal{H} = \mathcal{E} \times \mathcal{R} \times \mathcal{E} \backslash \mathcal{O}$, and let $\{(F_q, W_q)\}_{q=1}^m$ be a set of weighted rules, where F_q is a rule (see section 3.2.1 for details, where a rule is referred to as first-order logic), W_q is the corresponding rule weight. The knowledge graph completion task aims to infer the assignment vector for all unobserved facts $\boldsymbol{y} \in \mathbb{R}^{|\mathcal{H}|}$ given the observed facts and the rules.

3.2 Preliminaries

3.2.1 First-order logic

First-order logic. A first-order logic (also referred to as "logic rule" in this thesis) associated with a knowledge base $\mathcal{K}$ is an expression based on relations in $\mathcal{K}$. Formally, a logic rule F_q in clausal form can be represented as a disjunction of atoms and negated atoms:

$$\big(\vee_{i \in \mathcal{I}_q^+} r_i(A_i, B_i) \big) \vee \big(\vee_{i \in \mathcal{I}_q^-} \neg r_i(A_i, B_i) \big), \tag{3.1}$$

where $\mathcal{I}_q^-$ and $\mathcal{I}_q^+$ are two index sets containing the indices of atoms that are negated or not, respectively, A_i and B_i are variables. Logic rules in clausal form can be equivalently reorganized as an implication from premises (negated) to conclusions (non-negated):

$$\wedge_{i \in \mathcal{I}_q^-} r_i(A_i, B_i) \Longrightarrow \vee_{i \in \mathcal{I}_q^+} r_i(A_i, B_i). \tag{3.2}$$

The implication Eq. equation (3.2) is quite expressive since it includes many commonly used types of logic rule, e.g., symmetry/asymmetry, inversion, sub-relation, composition, etc.

3.2.2 Rule grounding and distance to satisfaction

Rule grounding. For a logic rule F_q in Eq. equation (3.2), by assigning entities $e \in \mathcal{E}$ to the variables A_i and B_i, F_q is *grounded*, producing a set of *ground for-*

mulas. For example, consider a simple logic rule $Father(A, B) \wedge Wife(C, A) \Rightarrow Mother(C, B)$. Let $A = Jack$, $B = Ross$ and $C = Judy$, we get a ground formula: $Father(Jack, Ross) \wedge Wife(Judy, Jack) \Rightarrow Mother(Judy, Ross)$.

Distance-to-satisfaction. Denote all ground formulas created by the q-th logic rule F_q by $\{G_q^{(j)}\}_{j=1}^{n_q}$, where n_q is the number of ground formula for the q-th rule. For any ground formula $G_q^{(j)}$ of F_q, when $r_i(h_i, t_i) \in \{0, 1\}$, the satisfaction of $G_q^{(j)}$ can be evaluated via Eq. equation (3.1). The value is either 1 or 0, meaning that the ground formula is satisfied or violated, respectively. The binary value of $r_i(h_i, t_i)$ can be relaxed to a continuous value ranging over $[0, 1]$. In such case, we may define the *distance-to-satisfaction* for $G_q^{(j)}$ via Łukasiewicz t-norm:

$$\mathrm{d}(G_q^{(j)}) \;\coloneqq\; \max\left\{1 - \sum_{i \in \mathcal{I}_q^+} r_i(h_i, t_i) - \sum_{i \in \mathcal{I}_q^-} (1 - r_i(h_i, t_i)),\; 0\right\}. \tag{3.3}$$

Note that $d(G_q^{(j)}) \in [0, 1]$, and the smaller d is, the better satisfied the ground formula $G_q^{(j)}$ is.

3.3 Differentiable neuro-symbolic reasoning

In this section, we propose a neuro-symbolic model, namely, DiffLogic, which unifies KG-embedding and rule-based reasoning. What follows we present the overall framework of DiffLogic, then show how to perform DiffLogic efficiently.

3.3.1 Overall framework

As shown in fig. 4.1, the model comprises three components: 1) an efficient grounding technique serving as a filter to identify crucial ground formulas and extract triples connected to them; 2) a KG-embedding model to compute truth scores for the extracted triples; and 3) a tailored continuous MLN framework that takes the truth

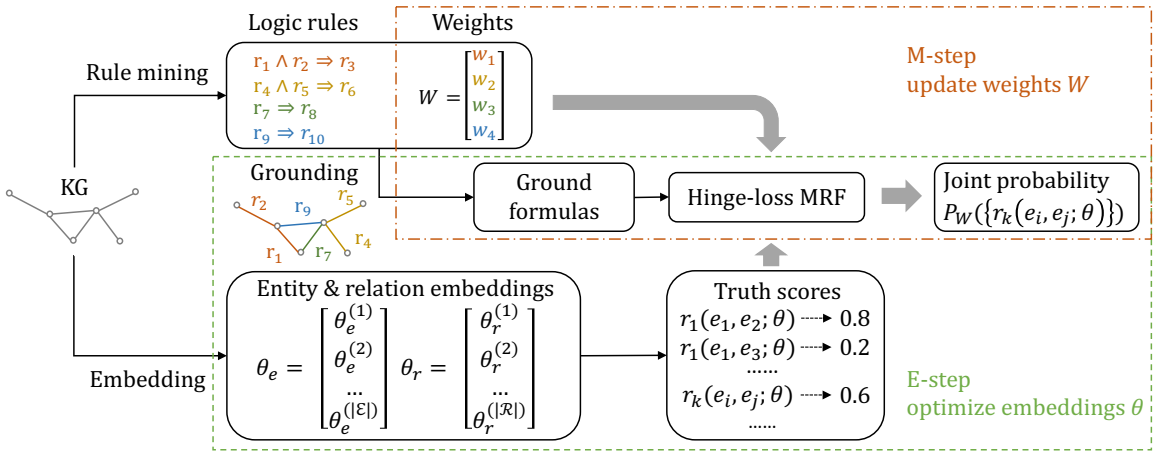

Figure 3.1: Overall framework of DiffLogic.

scores as input and assess the overall probability. The model is optimized using an EM algorithm, alternating between embedding optimization and weight updating. During the E-step, we fix the rule weights and optimize embeddings in an end-to-end fashion, by maximizing the overall probability; while in the M-step, we design an efficient rule weight updating method by leveraging the sparsity of violated rules. It is also worth mentioning here that the model requires a set of rules, which can be obtained from certain rule-mining process or domain experts.

Next, we turn to the abovementioned three components.

Rule-guided iterative grounding. The success of probabilistic logic reasoning heavily depends on the grounding process. However, the number of grounding formulas is overwhelmingly large, under our setting, equals $\sum_{q=1}^{m} |\mathcal{E}|^{|\mathcal{I}_q^-|+1}$. As a result, one has to sample for approximation. In this work, we propose a grounding technique called Rule-guided Iterative Grounding (RGIG) that incrementally identifies crucial ground formulas, reducing the number of ground formulas needed for optimization without compromising performance.

Inference inherently promotes the agreement between assignments (i.e., $\boldsymbol{y}$, $\boldsymbol{x}$) and the weighted rule set (i.e., $\{(F_q, W_q)\}_{q=1}^{m}$) by penalizing violated rules. Consequently, ground formulas with higher distance-to-satisfaction are more valuable in guiding

optimization. In light of this, we only need to find the ground formulas whose premise atoms in Eq. equation (3.2) have (or potentially have) high scores. Pursuing this idea, RGIG iteratively grounds all logic rules and updates a fact set $\mathcal{V}$. The fact set $\mathcal{V}$ is initialized with the observed facts $\mathcal{O}$ from the training set. In each iteration, rules are grounded only from the facts in $\mathcal{V}$ that match the premise parts of the rules. New facts are derived from the conclusion parts of the rules and subsequently added to $\mathcal{V}$. In the subsequent iterations, the updated $\mathcal{V}$ is used to derive more new facts. From our numerical experience, a few iterations of RGIG are sufficient, say 3. In the end, RGIG yields a set of ground formulas.

This grounding technique leverages the sparsity of violated rules (the distances to satisfaction for most ground formulas are approximately zero) to efficiently identify important ground formulas, facilitating data-efficient optimization. The case study in section 3.4.2 illustrates that our grounding technique attains comparable reasoning performance and is orders of $(10^3 \sim 10^5)$ more efficient in terms of the number of ground formulas.

KG-embedding model. In the literature of probabilistic logic reasoning, the representation of assignment $\boldsymbol{y}$ for all unobserved facts is very expensive, which is approximate $|\mathcal{E}|^2|\mathcal{R}|$ as the observed fact triples are only a small portion. To this end, we employ a KG-embedding model to parameterize $\boldsymbol{y}$ and $\boldsymbol{x}$. By embedding each entity to continuous representation with n_e parameters, and each relation with n_r parameters. The number of parameters is reduced from $|\mathcal{E}|^2|\mathcal{R}|$ to $|\mathcal{E}|n_e + |\mathcal{R}|n_r$, which is linear with respect to the number of entities/relations.

Although (re)-parameterization improves the representation efficiency, we also need to consider effectiveness. The relation pattern modeling capability is essential for honestly modeling the logic in a KG and performing reasoning. For example, the ability to model symmetric and reverse relation is attributed to capturing logic rules with one premise atom (e.g. $Husband(A, B) \Rightarrow Wife(B, A)$), while the ability to model composition relation is attributed to capturing logic rules with two premise

atoms (e.g. $Father(A, B) \wedge Wife(C, A) \Rightarrow Mother(C, B)$). Among all candidate KG-embedding models, we choose RotatE [49] for its simplicity and capability of modeling the various logic patterns. And of course, one may use also other KGE models instead. Let the KGE model be parameterized by θ. Using the sigmoid function, we can transform the score for the fact triple (h, r, t) produced by the KGE model into $[0, 1]$, which can be taken as the truth value for (h, r, t). So the truth value for (h, r, t) can be parameterized by θ. Therefore, the assignment vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ can be parameterized by θ, with their entries being the truth values for the observed and unobserved fact triples, respectively.

Hinge-loss Markov random field. Given the parameterized assignments, we employ PSL [1] to build a hinge-loss Markov random field (HL-MRF). Specifically, given a knowledge base $\mathcal{K}$ with assignments $\boldsymbol{x}, \boldsymbol{y}$, and a set of weighted logic rules $\{(F_q, W_q)\}_{q=1}^m$. A HL-MRF P over $\boldsymbol{y}$ conditioned on $\boldsymbol{x}$ is a probability density function defined as

$$P_{\boldsymbol{w}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{W}, \boldsymbol{x})} \exp(-f_{\boldsymbol{w}}(\boldsymbol{y}, \boldsymbol{x})), \qquad Z(\boldsymbol{W}, \boldsymbol{x}) = \int_{\boldsymbol{y}} \exp(-f_{\boldsymbol{w}}(\boldsymbol{y}, \boldsymbol{x}))\mathrm{d}\boldsymbol{y}, \quad (3.4)$$

where $f_{\boldsymbol{w}}$ is the hinge-loss energy function, defined as the weighted sum of all potentials:

$$f_{\boldsymbol{w}}(\boldsymbol{y}, \boldsymbol{x}) = \boldsymbol{W}^\top \boldsymbol{\Phi}(\boldsymbol{y}, \boldsymbol{x}) = [W_1, \ldots, W_m][\Phi_1(\boldsymbol{y}, \boldsymbol{x}), \ldots, \Phi_m(\boldsymbol{y}, \boldsymbol{x})]^\top = \sum_{q=1}^{m} W_q \Phi_q(\boldsymbol{y}, \boldsymbol{x}), \quad (3.5)$$

and $\Phi_q(\boldsymbol{y}, \boldsymbol{x})$ is the sum of potentials of all ground formulas of F_q, i.e., $\Phi_q(\boldsymbol{y}, \boldsymbol{x}) = \sum_{j=1}^{n_q} \mathrm{d}(G_q^{(j)})$.

HL-MRF optimizes the assignment $\boldsymbol{y}$ and the rule weights $\boldsymbol{W}$ by alternating between Maximum a posterior (MAP) inference and weight learning. The former step fixes the $\boldsymbol{W}$ and optimizes $\boldsymbol{y}$, while the latter step fixes $\boldsymbol{y}$ and updates $\boldsymbol{W}$.

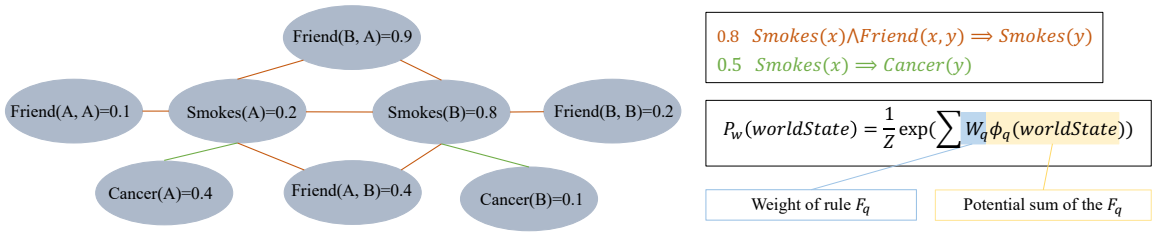

Figure 3.2: An illustration of probabilistic soft logic (PSL).

3.3.2 Optimization

Below we elaborate on the optimization details of DiffLogic. We demonstrate how the model optimization is efficiently performed by employing numerical optimization techniques and approximation methods that leverage sparse properties.

Embedding updating The task in the inference step is to infer $P_{\boldsymbol{w}}(\boldsymbol{y}|\boldsymbol{x})$, i.e., finding the optimum $\boldsymbol{y}$ given the observed assignment $\boldsymbol{x}$ and current weights $\boldsymbol{W}$, which can be formulated as a maximum a posterior (MAP):

$$\operatorname{argmax}_{\boldsymbol{y}} P_{\boldsymbol{w}}(\boldsymbol{y}|\boldsymbol{x}) \equiv \operatorname{argmin}_{\boldsymbol{y}\in[0,1]^{|\mathcal{H}|}} \boldsymbol{W}^{\top}\boldsymbol{\Phi}(\boldsymbol{y},\boldsymbol{x}).$$

To be consistent with the observation, $\boldsymbol{x}(\theta)$ should be as small as possible. Additionally, we also want the truth values for negative samples to be large. So, the overall cost function can be formulated as:

$$\min_{\theta} \boldsymbol{W}^{\top}\boldsymbol{\Phi}(\boldsymbol{y}(\theta),\boldsymbol{x}(\theta)) + \lambda\left(\frac{1}{|T^{+}|}\sum_{(h,r,t)\in T^{+}}[1-r(h,t;\theta)] + \frac{1}{|T^{-}|}\sum_{(h,r,t)\in T^{-}} r(h,t;\theta)\right), \tag{3.6}$$

where $\boldsymbol{x}$, $\boldsymbol{y}$ are parameterized by θ, $\lambda > 0$ is a parameter, $r(h,t;\theta) \in [0,1]$ stands for the truth value of a fact triple (h,r,t), T^{+} and T^{-} are positive and negative sample sets, respectively. The optimization problem Eq. equation (3.6) can be solved by stochastic gradient descent, where we sample a batch of ground formulas to compute the potentials, T^{+} is a batch of positive triples from $\mathcal{O}$, and T^{-} is a set of negative samples obtained by corrupting T^{+}.

Rule weights updating In this step, θ is fixed, and rule weights are updated by maximizing $\log P_{\boldsymbol{w}}(\boldsymbol{y}|\boldsymbol{x})$. The gradient of $\log P_{\boldsymbol{w}}(\boldsymbol{y}|\boldsymbol{x})$ with respect to the weight of F_q can be given by

$$\frac{\partial \log P_{\boldsymbol{w}}(\boldsymbol{y} \mid \boldsymbol{x})}{\partial W_q} = \mathbb{E}_{\boldsymbol{W}}\left[\Phi_q(\boldsymbol{y}, \boldsymbol{x})\right] - \Phi_q(\boldsymbol{y}, \boldsymbol{x}). \tag{3.7}$$

The first term computes the expectation of potential, which involves integration over $\boldsymbol{y}$ under the distribution defined by rule weight $\boldsymbol{W}$. Directly computing the integration is impractical, so we instead optimize the pseudo-likelihood of training data:

$$P^*_{\boldsymbol{w}}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{i=1}^{n} \frac{\exp\left[-f^i_{\boldsymbol{w}}\left(y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x}\right)\right]}{Z_i(\boldsymbol{W}, y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x})}, \quad f^i_{\boldsymbol{w}} = \sum_{q=1}^{m} W_q \sum_{j=1}^{n_q} \mathbf{1}_{\{y_i \to G_q^{(j)}\}} d(G_q^{(j)}).$$

Here $\mathbf{1}_{\{y_i \to G_q^{(j)}\}} = 1$ if y_i is connected to ground formula $G_q^{(j)}$, otherwise, zero. Z_i is the integration of $f^i_{\boldsymbol{w}}$ over y_i, similar to the second equation in Eq. equation (3.4). The pseudo-likelihood is a mean-field approximation for the exact likelihood in Eq. equation (3.4). It approximates the exact likelihood by decomposing it into a product of the probabilities of y_i's, conditioned on the Markov Blanket of y_i denoted by $\mathrm{MB}(y_i)$. The gradient of the pseudo-likelihood is as follows (see Appendix A.2.1 for a detailed derivation):

$$\frac{\partial \log P^*_{\boldsymbol{w}}(\boldsymbol{y} \mid \boldsymbol{x})}{\partial W_q} = \sum_{i=1}^{n} \left\{\mathbb{E}_{y_i|\mathrm{MB}}\left[\Psi_{q,MB(i)}\right] - \Psi_{q,MB(i)}\right\}, \ \Psi_{q,MB(i)} = \sum_{j=1}^{n_q} \mathbf{1}_{\{y_i \to G_q^{(j)}\}} d(G_q^{(j)}).$$

The above equality enables us to estimate gradients by minibatch sampling. For each sampled y_i, we need to compute both $\Psi_{q,MB(i)}$ and its expectation. The integration term can be estimated using Monte Carlo integration by fixing other variables and sampling y_i on the interval [0, 1]. The Monte Carlo integration is parallelizable and converges quickly as the number of samples increases.

To further reduce the computation burden, we leverage the sparsity of the violated ground formulas to filter ground formulas. Practically, we pre-compute the truth scores of all triples connected to the ground formulas using the KG-embedding model, then use a threshold to divide these triples into a "positive" set and a "negative"

set. A ground formula is expected to be violated if all its premise atoms are positive and all its conclusion atoms are negative. By only involving the violated rules when computing potentials, we significantly reduce computational costs.

Joint reasoning with rules and embeddings After training, we get the trained embeddings and updated rule weights, which can be used to perform reasoning. There are two ways to compute the score: 1) compute each embedding score $r_i(h_i, t_i; \theta)$ using the learned embeddings θ; 2) compute the cumulative rule score $f_{rule}(h_i, r_i, r_i; \boldsymbol{W})$ by summing up weights of ground formulas that infer the target triple:

$$f_{rule}(h_i, r_i, t_i; \boldsymbol{W}) = \sum_{q=1}^{m} W_q \sum_{j=1}^{n_q} \mathbf{1}_{\{r_i(h_i,t_i) \text{ can be inferred by } G_q^{(j)}\}}.$$

To perform joint inference using both embedding scores and rule scores, a simple way is to use the weighted sum of embedding scores and the normalized rule scores:

$$r_i(h_i, t_i) = (1 - \eta) \cdot r_i(h_i, t_i; \theta) + \eta \cdot \widehat{f}_i(h_i, t_i; \boldsymbol{W}), \tag{3.8}$$

where $\widehat{f}_i(h_i, t_i; \boldsymbol{W})$ is the rescaled value of $f_{rule}(h_i, r_i, t_i; \boldsymbol{W})$ calculated by minmax normalization. The optimum weight coefficient η is selected by using the validation set.

3.4 Experiments

In this section, we conduct experiments to answer the following research questions. **RQ1:** Can DiffLogic outperform rule-based and embedding-based models in terms of reasoning performance? **RQ2:** Can DiffLogic scale to large knowledge graphs that rule-based models struggle to handle? **RQ3:** Does DiffLogic actually learn representations compatible with rules? **RQ4:** Quantify the efficiency and scalability of DiffLogic. **RQ5:** How effective is the model in terms of leveraging prior knowledge encoded in rules, compared with data-driven methods?

Datasets and candidate rules. We incorporate four real-world knowledge graph datasets: YAGO3-10, WN18, WN18RR, and CodeX (available in three sizes: small, medium, and large), along with a synthetic logic reasoning dataset: Kinship. Dataset statistics and descriptions can be found below. We also list the statistics of the real-world knowledge graph datasets in Table 3.1 and the synthetic Kinship dataset in Table 3.2. We present detailed descriptions for each dataset below. Candidate rules for knowledge graphs are mined using AMIE3 [25], with rule weights initialized by rule confidence scores.

- **CodeX.** The CodeX dataset, recently proposed for knowledge graph completion tasks, is a comprehensive collection extracted from both Wikidata and Wikipedia. This challenging dataset comes in three versions: small (S), medium (M), and large (L), allowing for comprehensive evaluation.
- **YAGO3-10.** YAGO3-10 is a subset of YAGO3 [45], a large knowledge base completion dataset, with the majority of triples describing attributes of persons, including their citizenship, gender, and profession.
- **WN18.** WordNet 18 (WN18) dataset is one of the most commonly used subsets of WordNet.
- **WN18RR.** WN18RR is a modified version of WN18 designed to be more challenging for knowledge graph reasoning algorithms by removing reverse relations in the knowledge graph.
- **Kinship.** A synthetic dataset, widely used [66, 16] for evaluating the statistical relational learning ability and the scalability of reasoning algorithms. We use five different sizes of the dataset for evaluating its run time efficiency and parameter scalability, namely Kinship-S1/S2/S3/S4/S5, respectively.

Baseline models. Baseline models include four KG-embedding models — TransE [3], RotatE [49], TuckER [2], MLP [15], two GNN-based models — SACN [41], CompGCN

Table 3.1: Statistics of real-world knowledge base datasets.

Dataset	**#Ent**	**#Rel**	**#Train/Valid/Test**	**#Rules**
CodeX-s	2,034	42	32,888/1,827/1,828	35
CodeX-m	17,050	51	185,584/10,310/10,311	52
CodeX-l	77,951	69	551,193/30,622/30,622	57
YAGO3-10	123,182	37	1,079,040/5,000/5,000	22
WN18	40,943	18	141,442/ 5,000/ 5,000	140
WN18RR	40,943	11	86,835/ 3,034/ 3,134	51

Table 3.2: Statistics for Kinship datasets of varied sizes (S1-S5).

	S1	S2	S3	S4	S5
Number of rules containing 1 premise atom	12	12	12	12	12
Number of rules containing 2 premise atoms	9	9	9	9	9
Number of predicates	15	15	15	15	15
Number of entities	52	106	158	202	267

[52], four rule-learning models — Neural LP [59], DRUM [39], RNNLogic [35], RLogic [10], a discrete MLN engine MLN4KB [16], and a neuro-symbolic model pLogicNet [36] that also integrate KG-embedding and MLN. We exclude the recently proposed ExpressGNN [66] from our experiments on KG experiments since it requires querying test data[1] during training and is inapplicable in our setting. For all baselines that employ a KG embedding model, we unify the negative sampling scheme as adversarial negative sampling for fair comparison. Hyperparameters for each baseline are taken from their original paper.

[1]Please see the discussion on the usage of test data at `https://openreview.net/forum?id=rJg76kStwH`.

Table 3.3: Reasoning performance on real-world datasets. [T=n] means the maximum body length of mined rules is n. The best results are shown in bold and the second best are underlined. [NA] indicates that the model cannot finish inference within ten hours.

	CodeX-s		**CodeX-m**		**CodeX-l**		**WN18RR**		**YAGO3-10**	
	MRR	hit@10	MRR	hit@10	MRR	hit@10	MRR	hit@10	MRR	hit@10
MLP	0.279	0.502	0.197	0.347	0.190	0.339	0.139	0.218	0.365	0.575
RotatE	0.421	0.634	0.325	0.466	0.319	<u>0.453</u>	0.469	0.566	0.495	0.670
TuckER	0.444	0.638	0.328	0.458	0.309	0.430	0.470	0.526	-	-
TransE	0.353	0.607	0.320	0.481	0.308	0.452	0.218	0.510	0.436	0.647
SACN	-	-	-	-	-	-	0.470	0.540	-	-
CompGCN	-	-	-	-	-	-	0.479	0.546	-	-
AMIE	0.195	0.283	0.063	0.095	0.026	0.029	0.36	0.485	0.25	0.343
NeuraLP	0.290	0.395	NA	NA	NA	NA	0.433	0.566	NA	NA
DRUM(T=2)	0.290	0.393	NA	NA	NA	NA	0.434	0.565	NA	NA
DRUM(T=3)	0.342	0.542	NA	NA	NA	NA	0.486	0.586	NA	NA
RNNLogic$^+$	-	-	-	-	-	-	<u>0.51</u>	<u>0.597</u>	NA	NA
RLogic$^+$	-	-	-	-	-	-	**0.52**	**0.604**	**0.53**	**0.703**
MLN4KB	0.082	0.134	0.035	0.045	0.028	0.032	0.368	0.374	0.460	0.525
pLogicNet	0.342	0.505	0.306	0.448	0.270	0.388	0.440	0.534	0.387	0.595
DiffLogic	<u>0.445</u>	<u>0.662</u>	<u>0.335</u>	<u>0.487</u>	<u>0.326</u>	0.448	0.493	0.585	0.503	0.673
DiffLogic$^+$	**0.458**	**0.655**	**0.343**	**0.495**	**0.337**	**0.46**	0.50	0.587	<u>0.513</u>	<u>0.674</u>

3.4.1 Reasoning on real-world knowledge graphs

To answer **RQ1**, **RQ2**, and **RQ3**, we conduct link prediction tasks on several real-world datasets. Including WN18RR, YAGO3-10, and CodeX of three sizes (denoted as CodeX-s/m/l). Note that YAGO3-10 and Codex-l are large knowledge graphs that are suitable for assessing scalability. We report the performance of DiffLogic under

two settings, using embedding scores for reasoning (denoted by DiffLogic) and using both embedding scores and rule scores for reasoning (denoted by DiffLogic$^+$). We evaluate the performance using the Mean Reciprocal Rank (MRR) and Hit@10, with the results presented in Table 3.3. We also investigate the evolution of violated rules and MRR on WN18RR during the inference of DiffLogic and RotatE, as demonstrated in Figure 3.3. These results lead to several key observations:

First, DiffLogic surpasses both rule-based and KG embedding-based methods. This can be primarily attributed to the fact that it combines the ability of both sides: 1) the ability to utilize embeddings to model similarities among entities, which enhance the reasoning performance; and 2) compared with data-driven KG-embedding models that only learn rule patterns from a large amount of labeled data, DiffLogic can explicitly leverage rule patterns within a principled logic reasoning framework, leading to better overall performance. Our results also indicate that jointly using rules and learned embeddings for reasoning is more effective than solely relying on embeddings. This suggests that rules and embeddings can complement each other during the reasoning process, ultimately leading to more accurate and robust inferences.

Second, DiffLogic outperforms pLogicNet, a neuro-symbolic model. Although pLogicNet also integrates MLN and KG-embeddings for reasoning, the key difference is that DiffLogic optimizes MLN and KG-embedding using a unified objective, whereas pLogicNet accommodates the discrete nature of MLN and essentially employs an MLN as a data augmentation technique to annotate additional facts for training KG embeddings. As a result, the optimization of MLN and KG embedding in pLogicNet is less consistent, and its performance is sensitive to the annotation threshold. Its performance is lower than TransE - its base KG embedding model on several large KGs (YAGO3-10/CodeX-l) because the annotated triples are mostly false positive. This highlights the advantages of DiffLogic in providing a more coherent and robust optimization process.

Third, DiffLogic demonstrates superior scalability compared to rule-based methods. DiffLogic is adept at efficiently scaling to expansive KG datasets like YAGO3-10 and CodeX-l. Conversely, rule-based methods, including NeuraLP, DRUM, RNNLogic, and MLN4KB, often encounter challenges when attempting to scale to such large KGs. An exception is RLogic, which is a scalable rule-learning model designed to mine complex and lengthy rules for reasoning. Remarkably, our model achieves comparable results by including only simply rules with a rule body length of ≤ 2.

Furthermore, DiffLogic proficiently learns representations that align with both KG-embeddings and rules. As the left subfigure in Figure 3.3 demonstrates, the number of violated rules during the inference (embedding learning) phase is initially sparse, rises rapidly, and eventually decreases with training progression. We attribute this to 1) the random initialization of embeddings, which assigns low truth scores to most triples, resulting in fewer initial violations; 2) the model's training phase, where truth scores for training set triples increase but rule patterns are not yet fully captured, causing a rapid rise in rule violations; 3) as training progresses, KG embeddings begin to encapsulate rule patterns, reducing the number of violations. Importantly, despite its base embedding model being RotatE, DiffLogic is more effective than pure data-driven RotatE in capturing rule patterns, ensuring a consistent decrease in rule violations. As depicted in the middle subfigure in Figure 3.3, DiffLogic achieves faster convergence in test MRR compared to RotatE.

3.4.2 Scalability of optimization

To answer **RQ4**, we assess the scalability of our methodology, focusing primarily on optimization efficiency, i.e., the efficacy of grounding. We employ Kinship, a widely used [66, 16] synthetic benchmark dataset, to evaluate the efficacy and efficiency of RGIG in identifying important ground formulas. This dataset includes a training set,

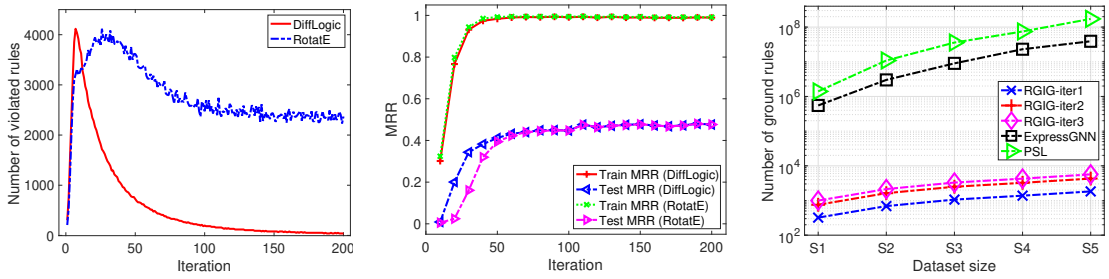

Figure 3.3: *Left*: Violated rules evolution during inference on WN18RR. *Middle*: MRR evolution on WN18RR. *Right*: Number of considered ground formulas in PSL/ExpressGNN and RGIG.

a test set, and a set of logic rules with full confidence. Some predicates are unobserved in the training set and can only be inferred via logic rules. The task is to deduce the gender of each individual in the test set, given the training set and the rule set. Kinship, designed for logical reasoning, necessitates resolving contradictions among ground formulas, thus the reasoning performance heavily depends on the grounding process. Such a challenging dataset is suitable for assessing the efficacy of grounding techniques.

In our empirical evaluations, we discovered that three iterations of RGIG are enough to identify crucial ground formulas for accurate reasoning, yielding an AUC-ROC of 0.982 ± 0.014 (detailed results are available in Section 3.4.4). We then compared the efficiency of RGIG with PSL and ExpressGNN's grounding techniques by examining the number of ground formulas considered for optimization across five different sizes of the Kinship dataset. Detailed calculations for PSL grounding can be found in Appendix A.3. The comparative results are depicted in the right subfigure in Figure 3.3. Although PSL and ExpressGNN can also perform accurate reasoning on Kinship, our model using RGIG has orders of $(10^3 \sim 10^5)$ more data-efficient optimization process over varied sizes of Kinship dataset, demonstrating its applicability.

To further evaluate the efficiency of RGIG, we empirically test the run-time and memory overhead on real-world datasets. Results are presented in Table 3.4.

Table 3.4: Grounding overhead on real-world datasets. Run-time overhead is evaluated ten times and report mean and std.

Datasets	CodeX-s	CodeX-m	CodeX-l	WN18RR	YAGO3-10
Run-time(/sec)	0.03±0.00	0.38±0.01	0.87±0.04	0.54±0.01	3.20±0.04
Memory(/MB)	2.19	11.57	25.58	18.73	262.65

The experimental results demonstrate that RGIG can efficiently scale to accommodate large KGs while maintaining minimal run-time and memory overhead. On the largest KG, YAGO3-10, the grounding process is completed in approximately 3.2 seconds, using around 262 MB of memory. In practice, we also observed that as the grounding iteration progresses, most inferred facts turn out to be negative triples. This trend could introduce noise and potentially impact efficiency. To enhance the efficiency of RGIG, one could consider filtering out facts with low scores by using a pre-trained RotatE model.

3.4.3 Learning from data vs. learning from rules

To answer **RQ5**, we design a "rule-pattern re-injection" experiment to evaluate DiffLogic's capability of injecting prior knowledge into embeddings, and compare it with pure data-driven based KG-embeddings. The design details are as follows:

The experiment contains two steps: 1) rule pattern removal, and 2) rule pattern re-injection. In our experiments, we use WN18 and select fourteen rules whose confidence scores are higher than 0.95. Then we deduplicate these rules so that no rules can be inferred from other rules, resulting in seven compact rules. In the **rule pattern removal step**, we use the selected seven rules to split the original training set, by finding all paths in the original training set that match the rules and extract the connected triples. The triples that match the conclusion part of the rules comprise the pattern set, and the original training set with the pattern set removed becomes the fact

set. In this way, the generated fact set does not contain any pattern for the seven rules. In the **rule pattern re-injection step**, we add different ratios $(0\%, 10\%, 20\%, 100\%)$ of the pattern set back into the fact set, so that rule patterns become increasingly evident. For DiffLogic, seven rules are applied to the training process for explicit rule pattern injection. We also include three KG-embedding models for comparison. To fairly compare the rule-injection capability for embeddings, we use only the embedding scores of DiffLogic for evaluation. Experimental results are presented in Table 3.5.

Table 3.5: Comparison of pure data-driven training and rule injection by DiffLogic for embeddings.

Model	MRR				Hits@10			
	0%	10%	20%	100%	0%	10%	20%	100%
MLP	0.123	0.156	0.198	0.851	0.279	0.364	0.449	0.932
TransE	0.399	0.469	0.500	0.775	0.917	0.944	0.944	0.957
RotatE	0.579	0.890	0.927	0.944	0.784	0.949	0.961	0.962
DiffLogic	**0.954**	**0.953**	**0.952**	**0.954**	**0.964**	**0.966**	**0.963**	**0.967**

The results show a significant difference between the two rule pattern learning paradigms: DiffLogic can directly leverage explicit prior knowledge compactly encoded in rules, achieving significant improvement in reasoning performance by using only a small number of logical rules. On the contrary, pure data-driven based KG-embedding algorithms can only implicitly learn rule patterns from labeled data. By adding more data from the pattern set back to the fact set, the performance of data-driven algorithms increases as the rule patterns become more evident in the knowledge base. Nevertheless, these data-driven algorithms are not comparable to DiffLogic even though rule patterns are observable.

Table 3.6: Comparative evaluation of reasoning performance on the Kinship dataset.

Algorithms	Ground iteration	AUC-ROC S1	S2	S3	S4	S5
PSL	-	.976±.011	.980±.005	.991±.003	.982±.005	.972±.004
ExpressGNN	-	.957±.002	.921±.001	.959±.004	.940±.001	.989±.004
DiffLogic-RotatE	1	.841±.005	.895±.001	.922±.001	.901±.001	.903±.000
	2	.931±.005	.994±.001	.998±.001	.985±.001	.993±.001
	3	.937±.005	.987±.001	.995±.001	.978±.001	.989±.001
DiffLogic-MLP	1	.567±.099	.537±.041	.507±.024	.503±.018	.504±.014
	2	.956±.032	.997±.002	.999±.003	.999±.001	**.999±.000**
	3	**.982±.014**	**.997±.001**	**.999±.001**	**.999±.000**	**.999±.000**

3.4.4 Probabilistic logic reasoning on Kinship Dataset

We assess performance on the Kinship dataset across five different sizes. Due to the full confidence of rules, we only perform inference in this experiment and do not need to update weights. We include DiffLogic using two different embedding models, i.e., RotatE and MLP, and evaluate their reasoning performance using RGIG with varied iterations (i.e., 1, 2, 3) for grounding. We include PSL and ExpressGNN as baselines, but we exclude pLogicNet due to its inability to utilize handcrafted rules. Given that the Kinship dataset lacks a validation set, we run each model ten times and report the AUC-ROC statistics from the final epoch of each run on the test set. The results are presented in Table 3.6, with the best results shown in bold.

3.4.5 Comparing inference time on Kinship

We evaluate the inference time on the Kinship dataset across five different sizes. We include models in Section 3.4.4 for this experiment. For two DiffLogic variants, we only evaluate their inference time when using 3 iterations of RGIG for grounding.

All the runtime experiments are conducted in the same machine with configurations as in Table 3.7. All of these models are implemented in Python, thereby ensuring a fair comparison. The inference time results are displayed in Table 3.8, with the best results shown in bold.

Table 3.7: Machine configuration.

Component	Specification
GPU	NVIDIA GeForce RTX 3090
CPU	Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz

Table 3.8: Comparison of runtime of inference on Kinship.

	Grounding	Runtime				
Algorithms	iteration	S1	S2	S3	S4	S5
PSL	-	∼3.6min	∼7.9min	∼12.9min	∼13.5min	∼32min
ExpressGNN	-	∼18.4min	∼19.1min	∼18.9min	∼19.4min	∼20.2min
DiffLogic-RotatE	3	37s	∼1.5min	∼3.2min	∼3.6min	∼4min
DiffLogic-MLP	3	**21.8s**	**41.5s**	**45s**	**54.4s**	**∼1.2min**

Chapter 4

Neuro-symbolic Entity Alignment via Variational Inference

4.1 Preliminaries

4.1.1 Problem statement

A knowledge graph $\mathcal{G}$ comprises a set of entities $\mathcal{E}$, a set of relations $\mathcal{R}$, and a set of relation triples $\mathcal{T}$ where each triple $(e_i, r_k, e_j) \in \mathcal{T}$ represents a directional relationship between its head entity and tail entity. Given two KGs $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, $\mathcal{G}' = \{\mathcal{E}', \mathcal{R}', \mathcal{T}'\}$, and a set of observed aligned entity pairs $\mathcal{O} = \{(e_i, e_i') | e_i \in \mathcal{E}, e_i' \in \mathcal{E}'\}_{i=1}^{n}$, the goal of entity alignment is to infer the missing alignments by reasoning with the existing alignments. This problem can be formulated in a probabilistic way: each pair $(e, e'), e \in \mathcal{E}, e' \in \mathcal{E}'$ is associated with a binary indicator variable $\boldsymbol{v}_{(e,e')}$. $\boldsymbol{v}_{(e,e')} = 1$ means (e, e') is an aligned pair, and $\boldsymbol{v}_{(e,e')} = 0$ otherwise. Given some observed alignments $\boldsymbol{v}_O = \{\boldsymbol{v}_{(e,e')} = 1\}_{(e,e') \in \mathcal{O}}$, we aim to predict the labels of the remaining hidden entity pairs $\mathcal{H} = \mathcal{E} \times \mathcal{E}' \backslash \mathcal{O}$, i.e., $\boldsymbol{v}_H = \{\boldsymbol{v}_{(e,e')}\}_{(e,e') \in \mathcal{H}}$.

4.1.2 Symbolic reasoning for entity alignment

Given an aligned pair (e_j, e'_j), a new aligned pair (e_i, e'_i) can be inferred with confidence score $w_{p,p'}$ if they are each connected to the existing pair via a relational path p and p' respectively, formally:

$$w_{p,p'}: \quad (e_j \equiv e'_j) \wedge p(e_i, e_j) \wedge p'(e'_i, e'_j) \Longrightarrow (e_i \equiv e'_i), \tag{4.1}$$

where $p = |\mathcal{R}|^L, p' = |\mathcal{R}'|^L$ are a pair of paths each consisting of L connected relations, and $w_{p,p'}$ measures the rule quality that considers the intra-KG structure and inter-KG structure, such as the indicative of each path, and the similarity between two paths. By instantiating such *rule* with the constants (real entities and relations) in the KG pair, a symbolic model predicts the label distribution of an entity pair (e, e') by:

$$p_w(\boldsymbol{v}_{(e,e')}|\mathcal{G}, \mathcal{G}'), \quad \text{for } (e, e') \in \{\mathcal{O} \cup \mathcal{H}\}. \tag{4.2}$$

Using logic rules to infer the alignment probability can leverage the high-order structural information for effective alignment as well as provide interpretability. However, exact inference is intractable due to the massive amount of possible instantiated rules (exponential to L), limiting its applicability to real-world KGs.

4.2 Neuro-symbolic reasoning framework for entity alignment

4.2.1 Variational EM

Given a set of observed labels $\boldsymbol{v}_O$, our goal is to maximize the log-likelihood of these labels, i.e., $\log p_w(\boldsymbol{v}_O)$. Directly optimizing this objective is intractable because it requires computing an integral over all the hidden variables. Instead, we optimize the evidence lower bound (ELBO) of the log-likelihood as follows:

$$p_w(\boldsymbol{v}_O) \geq E_{q(\boldsymbol{v}_H)}\left[\log p_w(\boldsymbol{v}_O, \boldsymbol{v}_H) - \log q(\boldsymbol{v}_H)\right] = \text{ELBO}(q, \boldsymbol{v}_O; w), \tag{4.3}$$

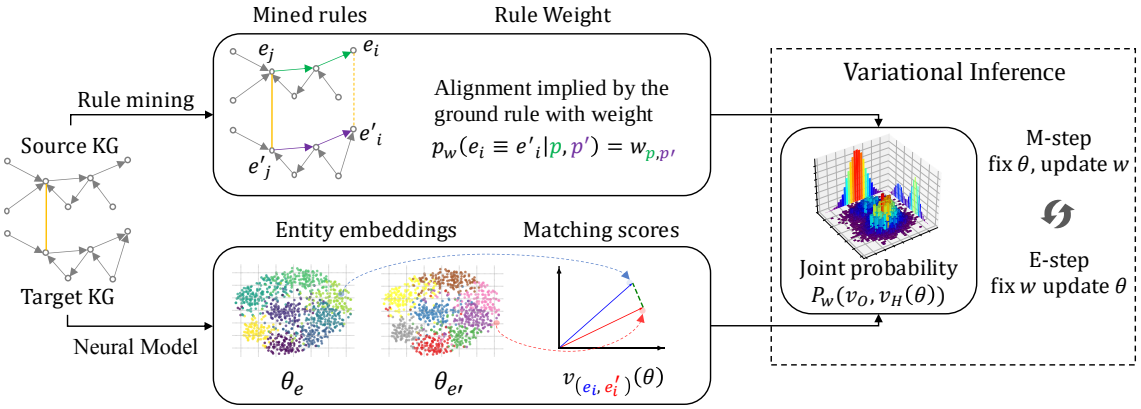

Figure 4.1: Framework illustration of NeuSymEA. The yellow solid line represents the alignment of an anchor pair. The symbolic model computes the matching probability of entity pairs by mining supporting rules (path pairs from anchor pairs) and evaluating their corresponding weights. The neural model learns embeddings and calculates entity-level matching scores based on embedding similarity. NeuSymEA models the agreement between the symbolic reasoning and neural representations using a joint probability distribution over observed pairs and parameterized truth scores for hidden pairs, optimized through a variational EM algorithm.

here, $q(\boldsymbol{v}_H)$ is a variational distribution of the hidden variables $\boldsymbol{v}_H$. This inequality holds for all q because $p_w(\boldsymbol{v}_O) = \text{ELBO}(q, \boldsymbol{v}_O; w) + D_{KL}(q(\boldsymbol{v}_H) \| p_w(\boldsymbol{v}_H \mid \boldsymbol{v}_O))$, where $D_{KL}(q(\boldsymbol{v}_H) \| p_w(\boldsymbol{v}_H \mid \boldsymbol{v}_O)) \geq 0$ is the KL-divergence between $q(\boldsymbol{v}_H)$ and $p_w(\boldsymbol{v}_H \mid \boldsymbol{v}_O)$. Under this framework, the log-likelihood $p_w(\boldsymbol{v}_O)$ can be optimized using an EM algorithm: during the E-step, we fix w and update the variational distribution q; during the M-step, we update w to maximize the log-likelihood of all the entity pairs, i.e., $E_{q(\boldsymbol{v}_H)}[\log p_w(\boldsymbol{v}_O, \boldsymbol{v}_H)]$, as illustrated in Figure 4.1.

Explicitly representing the variational distribution q is parameter intensive, which requires $\approx |\mathcal{E}||\mathcal{E}'|$ variables because the observed pairs are very sparse. To this end, we parameterize q with a neural model as q_θ, with θ being the parameters of the neural model.

4.2.2 E-step: inference

In this step, we fix w and update q_θ to minimize the KL divergence D_{KL}. Directly minimizing D_{KL} is intractable, as it involves computing the entropy of q_θ. Therefore, we follow [**?**] and optimize the reverse KL divergence of q_θ and p_w, leading to the following objective:

$$\phi_{\boldsymbol{v}_H,\theta} = \sum_{(e,e')\in\mathcal{H}} \mathbf{E}_{p_w(\boldsymbol{v}_{(e,e')}|\boldsymbol{v}_O)} q_\theta(\boldsymbol{v}_H). \tag{4.4}$$

To optimize this objective, we first use the symbolic model with weighted rules to predict $p_w(\boldsymbol{v}_{(e,e')} \mid \boldsymbol{v}_O)$ for each $(e, e') \in \mathcal{H}$. If $p_w(\boldsymbol{v}_{(e,e')} \mid \boldsymbol{v}_O) > \delta$, where δ is a threshold, we treat this entity pair as a positive label; otherwise, we regard the pair as a negative pair that can be selected during negative sampling process of the neural model.

The observed labels can also be used as training data for supervised optimization. The objective is:

$$\phi_{\boldsymbol{v}_O,\theta} = \sum_{(e,e')\in\mathcal{O}} \log q_\theta(\boldsymbol{v}_{(e,e')} = 1). \tag{4.5}$$

The final objective for q_θ is obtained by combining these two objectives: $\phi_\theta = \phi_{\boldsymbol{v}_H,\theta} + \phi_{\boldsymbol{v}_O,\theta}$.

4.2.3 M-step: Rule weight Update

In this step, we fix q_θ and update the rule weight w to maximize $\mathrm{ELBO}(q, \boldsymbol{v}_O; w)$. Since the right term of the ELBO in equation (4.3) is constant when q_θ is fixed, the objective is equivalent to maximizing the left term $E_{q_\theta(\boldsymbol{v}_H)}[\log p_w(\boldsymbol{v}_O, \boldsymbol{v}_H)]$, which is the log-likelihood function.

Specifically, we start by predicting the labels of hidden variables using the current neural model. For each $(e, e') \in \mathcal{H}$, we predict the labels $\hat{\boldsymbol{v}}_{(e,e')}(\theta)$ and obtain the prediction set $\hat{\boldsymbol{v}}_H(\theta) = \{\hat{\boldsymbol{v}}_{(e,e')}(\theta)\}_{(e,e')\in\mathcal{H}}$. In this way, maximizing the likelihood practically becomes maximizing the following objective:

$$\phi_w = \log p_w(\boldsymbol{v}_O, \hat{\boldsymbol{v}}_H(\theta)). \tag{4.6}$$

To obtain the pseudo-label $\hat{\boldsymbol{v}}_{(e,e')}$ using q_θ, we employ the trained neural model to compute the matching score of any entity pair $(e, e') \in \mathcal{H}$. However, this strategy can easily introduce false positives into the pseudo-label set especially when the number of entities is large. To mitigate this, we consider one-to-one matching to sift only the most confident pairs. Practically, we first sort all pairs by their confidence score, then we annotate the pairs as positive following the order of the confidence. If a pair contains an entity observed in the annotated pairs, then this pair is skipped. This simple greedy strategy significantly reduces the amount of false positives.

4.3 Optimization and Inference

4.3.1 Efficient optimization via logical deduction

Inference and learning with logic rules of length L can be computationally intensive, as the search space for paths grows exponentially with increasing L. To enhance reasoning efficiency, we decompose a rule in equation (4.1) using logic deduction, inspired by **(author?)** [8] in KG completion:

$$w_{p,p'}: \quad (e_j \equiv e'_j) \wedge \left(\bigwedge_{k=1}^{L} r_k(e_{k-1}, e_k)\right) \wedge \left(\bigwedge_{k=1}^{L} r'_k(e'_{k-1}, e'_k)\right) \implies (e_i \equiv e'_i). \quad (4.7)$$

Here $\bigwedge_{k=1}^{L} r_k(e_{k-1}, e_k)$ represents the path formed by $r_1, r_2, ..., r_L$ connecting e_i to e_j with $e_0 = e_i$ and $e_k = e_j$. This can be reorganized as *a series of single-step logic reasoning*:

$$w_{p,p'}: \quad (e_j \equiv e'_j) \wedge \left(\bigwedge_{k=1}^{L} \left[r_k(e_{k-1}, e_k) \wedge r'_k(e'_{k-1}, e'_k)\right]\right) \implies (e_i \equiv e'_i). \quad (4.8)$$

In this way, each logic rule of length L can be viewed as *a deductive combination of L short rules of length 1.* At each step, following [44], we perform one-step inference to update $p_w(\boldsymbol{v}_{(e,e')})$ for each $(e, e') \in \mathcal{H}$ by aggregating the alignment probability from neighbors:

$$1 - \prod_{\substack{(e,r,e_t)\in\mathcal{T},\\ (e',r',e'_t)\in\mathcal{T}'}} \left(1 - \eta(r) p_{sub}(r \subseteq r') p_w(\boldsymbol{v}_{(e_t,e'_t)})\right) \times \left(1 - \eta(r') p_{sub}(r' \subseteq r) p_w(\boldsymbol{v}_{(e_t,e'_t)})\right). \quad (4.9)$$

where $\eta(r)$ is a relation pattern of r measuring the uniqueness of e through relation r given a specified tail entity e_t, quantified by $\eta(r) = \frac{|\{e_t|(e_h,r,e_t)\in\mathcal{T}\}}{|\{(e_h,e_t)|(e_h,r,e_t)\in\mathcal{T})\}|}$. $p_{sub}(r \subseteq r')$ denotes the probability that relation r is a subrelation of r'. This formulation enables inference with confidence by explicitly quantifying confidence w using η and $p_{sub}(r \subseteq r')$. In this way, the update of the weight w simplifies to updating $p_{sub}(r \subseteq r')$ during the M-step (equation (4.6)), as $\eta(r)$ for each relation r is constant. In practice, the update

of $p_{sub}(r \subseteq r')$ can be computed by:

$$\frac{\sum \left(1 - \prod_{(e'_h, r', e'_t) \in \mathcal{T}'} \left(1 - \boldsymbol{v}_{(e_h, e'_h)} \boldsymbol{v}_{(e_t, e'_t)}\right)\right)}{\sum \left(1 - \prod_{e'_h, e'_t \in \mathcal{E}'} \left(1 - \boldsymbol{v}_{(e_h, e'_h)} \boldsymbol{v}_{(e_t, e'_t)}\right)\right)}. \tag{4.10}$$

where $\boldsymbol{v}_{(e_h, e'_h)}$ and $\boldsymbol{v}_{(e_t, e'_t)}$ are labels (or pseudo-labels) from $\boldsymbol{v}_O \cup \hat{\boldsymbol{v}}_H(\theta)$.

After optimization, rule weights can be computed by taking the product of the importance scores η of relations and the sub-relation probabilities of the corresponding relation pair:

$$w_{p,p'} \coloneqq \prod_{k=1}^{L} \eta(r_k) \cdot \eta(r'_k) \cdot \frac{p_{sub}(r_k \subseteq r'_k) + p_{sub}(r'_k \subseteq r_k)}{2}. \tag{4.11}$$

4.3.2 Inference with interpretability

To predict new alignments, there are two approaches: using the symbolic model or the neural model. The symbolic model infers alignment probabilities with the optimized weights w. Due to scalability concerns, symbolic methods generally adopt a lazy inference strategy that only preserves the confident pairs implied by the neighbor structure during inference. On the other hand, the neural model computes similarity scores for all entity pairs $(e, e') \in \mathcal{H}$ using the learned parameters θ, generating a ranked candidate list for each entity.

The evaluation of these models thus differs. Symbolic models are generally evaluated by precision, recall, and F1-score for their binary outputs, while neural models are assessed using hit@k and mean reciprocal ranks (MRR) for their ranked candidate lists. Following the practices in [34] and [30], we unify the evaluation metrics by treating the recall metric of symbolic models as equivalent to hit@1, facilitating comparison with neural models.

To enhance the interpretability of predictions, we adapt the optimized symbolic model into an explainer. For any given entity pair, the explainer generates a set

of supporting rule path pairs that justify their alignment, each associated with a confidence score indicating its significance. The explainer operates in two modes: **(1) hard-anchor mode**, which generates supporting paths only from prealigned pairs, and **(2) soft-anchor mode**, which includes paths from both prealigned and inferred pairs, providing more informative interpretations.

By integrating a breadth-first search algorithm (detailed in Appendix A.4.1), the explainer efficiently generates high-quality interpretations. For truly aligned pairs, it typically produces high-confidence interpretations, while for non-aligned pairs, the interpretations may result in an empty set (indicating no supporting evidence) or have low confidence scores. See Figure 4.3 for a visualized comparison.

4.4 Experiments

4.4.1 Experimental settings

In this section, we conduct experiments to answer the following questions. **RQ1:** Can NeuSymEA outperform existing neural, symbolic, and neuro-symbolic methods in terms of alignment performance? **RQ2:** Can symbolic and embedding-based methods complement each other in our framework? **RQ3:** How does the incorporated embedding-based model affect the alignment performance? **RQ4:** How does NeuSymEA interpret the inferred entity pairs, and how is the effectiveness of interpretations with respect to the rule length?

Datasets. We utilize the multilingual DBP15K dataset, which consists of three cross-lingual knowledge graph (KG) pairs: ja-en, fr-en, and zh-en. Note that the original full version [46] of this dataset contains a significant number of long-tail entities, presenting challenges for GCN-based models in terms of sparsity and large size. Therefore, many recent works [55, 33, 30] employ a condensed version, where long-tail entities and their

connected triples are removed. For a comprehensive evaluation, we use both versions. Detailed statistics for these datasets are provided in Section 4.4.1 and Section 4.4.1. Each dataset is divided into training, validation, and test sets following a 5-fold cross-validation scheme, with a ratio of 2:1:7.

Datasets	KG	Entities	Relations	Rel. Triplets	Aligned Entity Pairs
zh-en	Chinese (zh)	66,469	2,830	153,929	15,000
	English (en)	98,125	2,317	237,674	
ja-en	Japanese (ja)	65,744	2,043	164,373	15,000
	English (en)	95,680	2,096	233,319	
fr-en	French (fr)	66,858	1,379	192,191	15,000
	English (en)	105,889	2,209	278,590	

Table 4.1: Data statistics of the full DBP15K dataset.

Datasets	KG	Entities	Relations	Rel. Triplets	Aligned Entity Pairs
zh-en	Chinese (zh)	19,388	1,701	70,414	15,000
	English (en)	19,572	1,323	95,142	
ja-en	Japanese (ja)	19,814	1,299	77,214	15,000
	English (en)	19,780	1,153	93,484	
fr-en	French (fr)	19,661	903	105,998	15,000
	English (en)	19,993	1,208	115,722	

Table 4.2: Data statistics of the condensed DBP15K dataset.

Baselines and metrics. Baseline models include six neural models – GCNAlign [55], AlignE, BootEA [47], RREA [33], Dual-AMN [31], LightEA [32], one symbolic models – PARIS [44], and two neuro-symbolic models – PRASE [34], EMEA [30]. We use Hit@1, Hit@10, and MRR as the evaluation metrics. For PARIS and PRASE that have binary outputs, we report their recall as Hit@1, following **(author?)** [30].

Hyperparameters. NeuSymEA has two key hyperparameters: the number of EM iterations and the threshold δ for selecting positive pairs from the symbolic model. We tune these hyperparameters and select the best values based on the validation set. The search space for δ is $\{0.6, 0.7, 0.8, 0.9, 0.95, 0.98, 0.99\}$, while the number of iterations is searched from 1 to 9.

4.4.2 Results

Comparison with baselines

To answer **RQ1**, **RQ2** and **RQ3**, we compare NeuSymEA with baseline models on two versions of the DBP15K dataset: the full version (group1) and the condensed version (group2). Results are presented in Table 4.3. The results for PRASE, and EMEA on the condensed DBP15K are adopted from the original EMEA paper. We employ both Dual-AMN and LightEA as the neural models in our framework, denoted as NeuSymEA-D and NeuSymEA-L, respectively. These results lead to several key observations:

First, NeuSymEA surpasses both symbolic and KG embedding-based models. This can be attributed to the fact that it combines the capacity of both sides: 1) the ability to precisely infer confident pairs with rules by leveraging the multi-hop relational structures; and 2) the ability to effectively model entity representations in a unified space to jointly optimize the alignment of two KGs. NeuSymEA seamlessly combines both symbolic reasoning and neural representations in a principled variational framework, leading to improved performance.

Second, NeuSymEA outperforms both neuro-symbolic models, PRASE and EMEA. This improvement can be largely attributed to the model objective design in our framework. While PRASE and EMEA treat the symbolic and neural models as separate components, NeuSymEA integrates them within a principled variational

Table 4.3: Entity alignment results on DBP15K dataset. The suffixes "-D" and "-L" indicate the use of Dual-AMN and LightEA as the neural models. The results of RREA and EMEA are omitted on the full dataset due to an OOM (Out of Memory) error.

Category	Model	ja-en			fr-en			zh-en		
		Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR
Group 1: Results on the full DBP15K dataset										
Neural	GCNAlign	0.221	0.461	0.302	0.205	0.475	0.295	0.189	0.438	0.271
	BootEA	0.454	0.782	0.564	0.443	0.799	0.564	0.486	0.814	0.600
	AlignE	0.356	0.715	0.476	0.346	0.731	0.475	0.333	0.690	0.453
	Dual-AMN	0.627	0.883	0.717	0.652	0.908	0.744	0.650	0.884	0.732
	LightEA	0.736	0.894	0.793	0.782	0.919	0.832	0.725	0.874	0.779
symbolic	PARIS	0.589	-	-	0.618	-	-	0.603	-	-
Neuro-Symbolic	PRASE	0.611	-	-	0.647	-	-	0.652	-	-
Ours	NeuSymEA-D	**0.806**	**0.942**	**0.855**	0.827	**0.952**	**0.871**	**0.801**	**0.925**	**0.843**
	NeuSymEA-L	0.781	0.907	0.826	**0.834**	0.937	**0.871**	0.785	0.894	0.825
Group 2: Results on the condensed DBP15K dataset										
Neural	GCNAlign	0.331	0.662	0.443	0.325	0.688	0.446	0.335	0.653	0.442
	BootEA	0.530	0.829	0.631	0.579	0.872	0.961	0.575	0.847	0.668
	AlignE	0.433	0.783	0.552	0.457	0.821	0.580	0.474	0.806	0.587
	RREA	0.749	**0.935**	0.818	0.797	**0.958**	0.859	0.762	**0.938**	0.827
	Dual-AMN	0.750	0.927	0.815	0.793	0.954	0.854	0.756	0.919	0.816
	LightEA	0.778	0.911	0.828	0.827	0.943	0.830	0.770	0.894	0.816
symbolic	PARIS	0.565	-	-	0.584	-	-	0.543	-	-
Neuro-Symbolic	PRASE	0.580	-	-	0.622	-	-	0.593	-	-
	EMEA	0.736	-	0.807	0.773	-	0.841	0.748	-	0.815
Ours	NeuSymEA-D	0.805	0.930	0.849	0.835	0.953	0.879	**0.815**	0.926	**0.855**
	NeuSymEA-L	**0.811**	0.928	**0.854**	**0.858**	0.954	**0.894**	0.804	0.904	0.840

EM framework, unifying their objective as the log-likelihood of the observed variables. This approach allows for the joint optimization of both components, ensuring they work together to maximize the log-likelihood. By iteratively refining both components

through the EM algorithm, NeuSymEA achieves a more coherent and convergent solution, leading to superior performance, as shown by the empirical results.

Finally, NeuSymEA demonstrates superior robustness across both versions of the DBP15K dataset. Comparisons between two groups of results offer an interesting insight: embedding-based models experience significant performance degradation when moving from the condensed version to the full version of DBP15K (e.g., MRR of Dual-AMN decreases from 0.815 to 0.717 on ja-en), while symbolic models, in contrast, show improvements. We attribute this to two key factors: (1) Embedding-based models rely on entity-level matching, which is sensitive to dataset size. As the dataset grows, the number of similar entity embeddings increases, leading to reduced accuracy. The full version of DBP15K contains significantly more entities than the condensed version, exacerbating this effect. (2) Symbolic models, on the other hand, perform path-level matching. Their effectiveness is constrained more by substructure heterogeneity and sparsity than dataset size. The full version of DBP15K includes more relational triples, which enhances the rule-mining process, ultimately making the symbolic models more effective in this scenario. **NeuSymEA, by combining symbolic reasoning with KG embeddings, mitigates the shortcomings of both approaches, making it robust to changes in dataset scale and structure. This synergy allows NeuSymEA to consistently outperform baselines in both sparse and dense settings.**

Evolution of rules and embeddings

We study how rules and embeddings evolve and interact with each other during the EM steps, with results shown in Figure 4.2. Results in the left subplot indicate that in each EM iteration, the number of rule-inferred pairs grows consistently with high precision, implying that the embedding model continuously improves the inference performance of rules. These precise pairs, in turn, enhance the performance of the

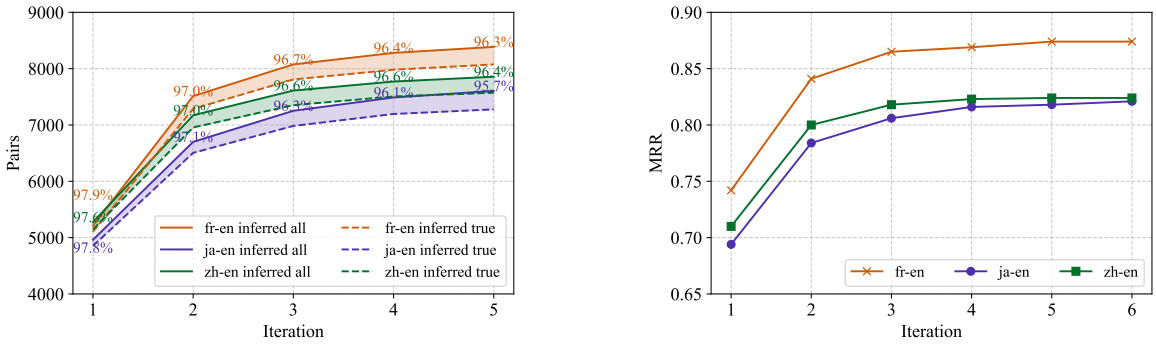

Figure 4.2: (Left) Evolution of rule inferred pairs, with solid lines representing total inferred pairs and dashed lines representing true inferred. The shaded areas indicate the number of false pairs. Precision values are annotated at each data point. (Right) Convergence of MRR of the neural model.

neural model. As shown in the right subfigure, the MRR of the neural model converges within a few iterations.

Interpretations by the Explainer

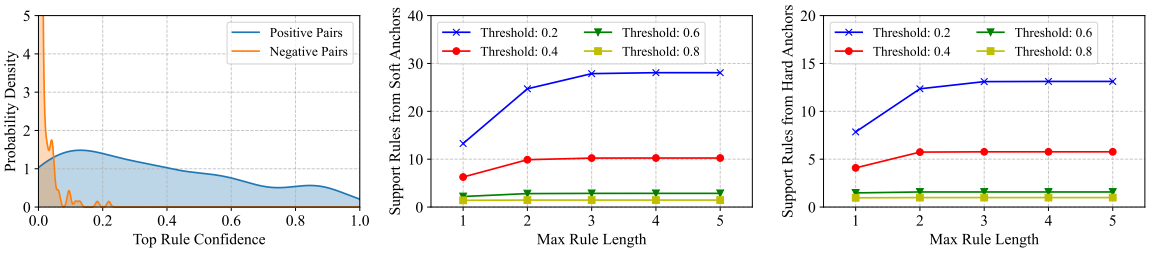

Figure 4.3: (Left) Probability density of the top supporting rule's confidence; (Middle) Number of supporting rules (thresholded by confidence score) relative to the maximum rule lengths under the **soft anchor mode**; (Right) Number of supporting rules relative to the maximum rule lengths under the **hard anchor mode**.

We investigate the interpretations generated by the explainer on the fr-en dataset. The left subfigure of Figure 4.3 shows the probability density of confidence scores for sup-

Table 4.4: Examples of supporting rules for query pairs in fr-en. Anchor pairs are shown in bold.

Query Pair	Supporting Rule	Confidence
Maison_de_Savoie House_of_Savoy	(Humbert_II_(roi_d'Italie), dynastie, Maison_de_Savoie), (Humbert_II_(roi_d'Italie), conjoint, **Marie-José_de_Belgique**) (Umberto_II_of_Italy, house, House_of_Savoy), (Umberto_II_of_Italy, spouse, **Marie_José_of_Belgium**)	0.80
Légion_espagnole Spanish_Legion	(Légion_espagnole, commandantHistorique, Francisco_Franco), (Francisco_Franco, conjoint, **Carmen_Polo**) (Spanish_Legion, notableCommanders, Francisco_Franco), (Francisco_Franco, spouse, **Carmen_Polo,_1st_Lady_of_Meirás**)	0.59
Premier_ministre_du_Danemark Prime_Minister_of_Denmark	(Premier_ministre_du_Danemark, titulaireActuel, **Lars_Løkke_Rasmussen**) (Prime_Minister_of_Denmark, incumbent, **Lars_Løkke_Rasmussen**)	0.79

porting rules (with a maximum rule length of 3) associated with entity pairs. Positive pairs are derived from the test set, while negative pairs are created by replacing one entity in each test pair with another randomly sampled entity. The distinct confidence distributions indicate that **positive pairs generally have more evidence for alignment**, which aligns with intuition. However, the probability density distribution also reveals that some positive pairs do not have high confidence scores. Upon further examination, we found that **many test pairs are isolated, i.e., they lack directly aligned neighbors. Despite this, NeuSymEA successfully generates supporting rules for isolated pairs by exploiting multihop dependencies**. In Table 4.4, we provide several examples of supporting rules and their associated confidence scores for the queried entity pairs.

To examine the impact of rule length on the explainer's effectiveness, the middle and right subfigures in Figure 4.3 show the number of supporting rules for test positive pairs as the maximum rule length increases. Compared to hard anchor mode, the explainer in soft anchor mode generates more high-quality supporting rules by leveraging inferred pairs as complementary anchor pairs, mitigating the sparsity issue. We also observe that **increasing the maximum rule length leads to more high-quality rules; however, the number of high-confidence rules grows more slowly than lower-confidence rules.** This can be attributed to our method for calculating confidence: the logical deduction-based approach computes a rule's confidence as the product of the confidences of its decomposed length-one sub-rules (as in Equation (4.11)). For

example, a rule with two sub-rules, each with confidence 0.8, results in an overall confidence of $0.8 \times 0.8 = 0.64$. Considering this, the confidence score tends to decrease when the rule length increases, thus increasing the maximum length tends to discover supporting rules with lower scores.

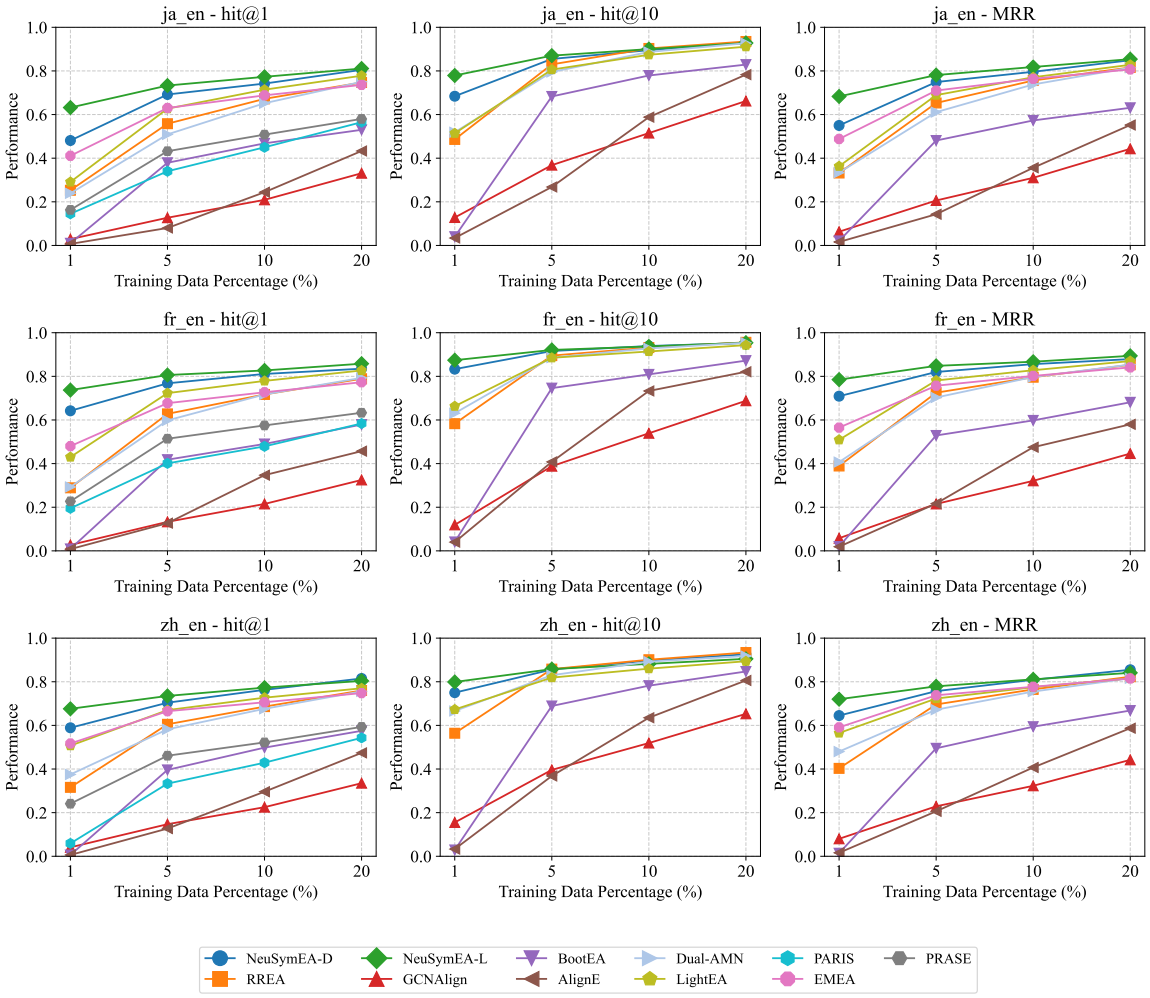

Figure 4.4: Alignment performance with varying percentages of pairs as training data.

Robustness in low resource scenario

Figure 4.4 illustrates the performance of various models in low-resource settings, where the amount of training data is significantly limited. As expected, all models experience

performance declines in Hit@1 as the percentage of training data decreases. However, several key observations stand out:

- **Robustness of NeuSymEA:** NeuSymEA (both NeuSymEA-L and NeuSymEA-D) demonstrates **remarkable robustness**, consistently outperforming other models across all datasets, even with minimal training data. Notably, when trained with only 1% of the data, NeuSymEA-L achieves a Hit@1 score above 0.7 on the fr-en dataset, surpassing some state-of-the-art models trained on 20% of the data. This highlights the effectiveness of combining neural and symbolic reasoning in entity alignment, offering a significant advantage over purely neural or symbolic methods.

- **Impact of Training Data:** While all models generally improve as the amount of training data increases, the **rate of improvement varies**:

  - Traditional neural models like GCNAlign and AlignE show steep improvements, indicating their heavy reliance on larger datasets.
  - In contrast, NeuSymEA maintains high performance even with minimal training data, showcasing its efficiency and potential for low-resource scenarios.

- **Language Pair Difficulty:** The experiments reveal varying degrees of difficulty in entity alignment across different language pairs:

  - **Japanese-English (ja-en)** alignment consistently achieves the highest performance across models.
  - **French-English (fr-en)** follows closely behind, while **Chinese-English (zh-en)** proves the most challenging.

  These differences may be influenced by factors such as linguistic distance, writing system differences, or the availability of pre-existing resources, highlighting the need for language-specific strategies in entity alignment.

- **Scalability and Adaptability of NeuSymEA:** The consistent performance of NeuSymEA across various datasets, metrics, and training data levels suggests that it is both **scalable** and **adaptable**. Its robustness makes it well-suited for cross-lingual knowledge integration tasks across diverse domains and languages. Additionally, its strong performance with limited data indicates its potential applicability in low-resource scenarios, allowing it to be deployed effectively in languages or domains with limited training data.

In summary, **NeuSymEA stands out as a robust and scalable solution** for entity alignment, excelling in low-resource settings and adapting well to diverse language pairs and contexts. This makes it a highly versatile model for cross-lingual knowledge integration tasks.

Parameter analysis

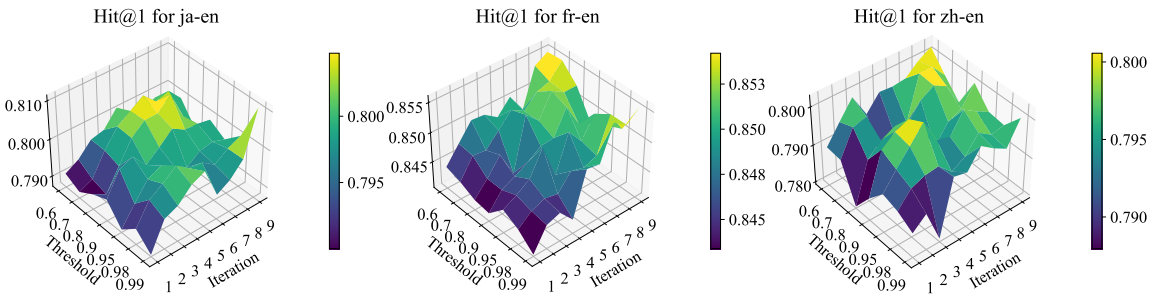

Figure 4.5: Performance sensitivity to hyperparameters iteration and threshold δ.

We present the hit@1 performance of NeuSymEA across three datasets, varying hyperparameters, illustrated by a three-dimensional graph. The threshold hyperparameter δ is explored within the set $\{0.6, 0.7, 0.8, 0.9, 0.95, 0.98, 0.99\}$, while the number of EM iterations ranges from 1 to 9. Performance levels are indicated using a colormap. Performance sensitivity analysis in Figure 4.5 reveals that for all datasets, performance generally improves as the iteration increases. On the other hand, the performance is less sensitive to the threshold δ.

Chapter 5

Probabilistic reasoning for zero-shot Entity Alignment with Large Language Models

5.1 Problem definition

A knowledge graph $\mathcal{G}$ comprises a set of entities $\mathcal{E}$, a set of relations $\mathcal{R}$, and a set of relation triples $\mathcal{T}$ where each triple $(e_h, r, e_t) \in \mathcal{T}$ represents a directional relationship between its head entity and tail entity. Given two KGs $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, $\mathcal{G}' = \{\mathcal{E}', \mathcal{R}', \mathcal{T}'\}$ and a fixed query budget $\mathcal{B}$ to a Large Language Model, we aim to train an entity alignment model θ based on the LLM's annotations to infer the matching score $m_\theta(e, e')$ for all entity pairs $\{(e, e'), e \in \mathcal{E}, e' \in \mathcal{E}'\}$. The evaluation process utilizes a ground truth alignment set $\mathcal{A}$ to assess the prediction accuracy for target entities in both directions, i.e.,$(e, ?)$ and $(?, e')$ for each true pair $(e, e') \in \mathcal{A}$, based on the ranked matching scores m_θ. Evaluation metrics are hit@k (where $k \in \{1, 10\}$) and mean reciprocal rank (MRR).

5.2 Entity alignment with noisy annotations from LLMs

We aim to design a framework to perform entity alignment with LLMs. Our design is motivated by the following insights. Firstly, we have a huge search space (the overall annotation space is $O(|\mathcal{E}||\mathcal{E}'|)$) to identify the core entity pairs to annotate. Secondly, we don't know whether the annotated labels are correct or not, because we have no prior knowledge or heuristic of the label distribution. Finally, we perform annotations iteratively, requiring the model to adjust its search policy based on annotation effectiveness, while we have no verifiable feedback of this annotation accuracy.

Based on these insights, we propose LLM4EA—an iterative framework that consists of four interconnected steps in each cycle, as illustrated in Figure 5.1. Initially, **an active selection** technique optimizes the use of resources by choosing critical source entities that significantly reduce uncertainty for themselves and their neighbors. Subsequently, **an LLM-based annotator** identifies the counterparts for the selected source entities, generating a set of pseudo-labels. Next, **a label refiner** improves label accuracy by eliminating structurally incompatible labels. This process involves formulating a combinatorial optimization problem and utilizing a probabilistic-reasoning-based greedy search algorithm to efficiently find a local-optimal solution. Finally, these refined labels are used to train **a base EA model** for the entity alignment task. The outcomes of the alignment then serve as feedback to inform subsequent rounds of the active selection policy. Further details are provided below.

5.2.1 Active selection of source entity

We aim to maximize the utility of the budget by actively allocating the budget to those beneficial entities. To do this, we sample source entities that reduce the most un-

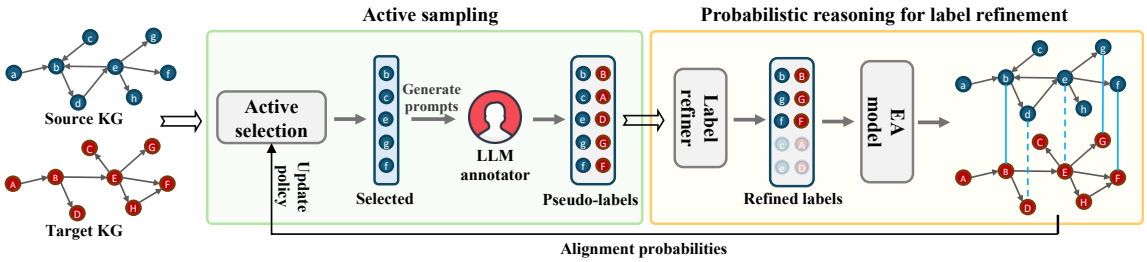

Figure 5.1: Overview of the LLM4EA framework. LLM4EA utilizes active sampling to select important entities based on feedback from an EA model. It also includes a label refiner to effectively train the base EA model using noisy pseudo-labels. Feedback from the EA model updates the selection policy.

certainty of both themselves and their neighboring entities, by a dynamically adjusted policy. The measurement of uncertainty reduction is based on two assumptions: 1) an entity's own uncertainty is inversely proportional to its alignment probability with its most probable counterpart; 2) the amount of uncertainty an entity eliminates for its neighbors is closely linked to the relational ties between them. To systematically assess this, we introduce the concept of *relational uncertainty*, quantified as follows:

$$U_r(e_h) = (1 - P(e_h)) + \sum_{(e_h, r, e_t) \in \mathcal{T}} w_r \left(1 - P(e_t)\right). \tag{5.1}$$

Here, w_r is a weight coefficient reflecting the significance of relation r and signifies how much e_h contributes to reducing the uncertainty of e_t through the relation r. For this purpose, we employ functionality $\mathcal{F}(r)$ (formally defined in Eq. equation (5.4)) as the weight w_r, as it quantifies the uniqueness of the tail entity for a given specified head entity. $P(e) \coloneqq \max_{e' \in \mathcal{E}'} P(e \equiv e')$ represents the alignment probability of the top-match entity for e. These alignment probabilities $P(e \equiv e')$ are obtained through probabilistic reasoning during label refinement (Section 5.2.3) and are augmented by the inferred alignments from the base EA model (Section 5.2.4). In the initial iteration, all alignment probabilities are set to 0.

It's important to note that some source entities are linked to a large number of uncertain neighbors (those with low $P(e)$). These source entities are crucial but may

be overlooked if their connected relations have low functionality. Hence, we introduce *neighbor uncertainty* as another metric to assess an entity's importance, by removing the functionality-based weight coefficient:

$$U_n(e_h) = (1 - P(e_h)) + \sum_{(e_h,r,e_t)\in\mathcal{T}} (1 - P(e_t)) \,. \tag{5.2}$$

To integrate these two metrics, we employ rank aggregation by mean reciprocal rank:

$$U(e_h) = 2 \times \left(\frac{1}{r_{ur}(e_h)} + \frac{1}{r_{un}(e_h)} \right) . \tag{5.3}$$

Here, $r_{ur}(e_h)$ and $r_{un}(e_h)$ denote the ranking of e_h when using U_r and U_n as metric, respectively. This simple-effective aggregation technique is advantageous for our task since it's scale invariant and requires no validation set for tuning hyperparameters, making it more practical in this task.

5.2.2 LLM as annotator

Counterpart filtering. With the selected source entities, we employ an LLM as an annotator to identify the counterpart from $\mathcal{E}'$ for each source entity, generating a set of pseudo-labels $\mathcal{L} = \{(e, e')|e \in \mathcal{E}, e' \in \mathcal{E}'\}$. To narrow down the search space, we first filter out the less likely counterparts before querying the LLM, selecting only the top-k most similar counterparts from $\mathcal{E}'$. The similarity metric is flexible: we use a string matching score based on word edit distance, but other methods are also viable, such as semantic embedding distances derived from word embedding models. By adjusting k, we can trade-off between the recall rate of counterparts and the query cost.

Prompt design. There are primarily two methods for retrieving context information to construct textual prompts: randomly generated prompts and dynamically tuned prompts. The former involves randomly selecting neighbors to construct contexts for the entity, while the latter dynamically selects neighbors based on feedback from the EA model. For a fair comparison, we use randomly generated prompts across all

baselines and the proposed LLM4EA. These prompts include the name of each entity and a set of relation triples to three randomly selected neighbors. For the baseline models, pseudo-labels are generated at once and used for training. For LLM4EA, we evenly divide the budget $\mathcal{B}$ into n iterations and generate pseudo-labels at each iteration using the allocated $\mathcal{B}/n$ budget.

5.2.3 Probabilistic reasoning for label refinement

The pseudo-labels generated by the LLM can be noisy, and directly using these labels to train an entity alignment (EA) model could undermine the final performance. Although estimating the label distribution by asking the LLM for confidence scores or querying multiple times to measure consistency are potential solutions, these approaches can be vulnerable or introduce additional costs.

In light of this, we propose a label refiner that leverages the structure of knowledge graphs. The refinement process is framed as a combinatorial optimization problem aimed at minimizing overall structural incompatibility among labels. Utilizing a probabilistic reasoning technique, we progressively update our confidence estimation for each label and select those that are mutually compatible, ultimately producing a set of accurate pseudo-labels. Detailed explanations follow below.

Functionality and probabilistic reasoning

Functionality. The functionality of a relation quantifies the uniqueness of tail entities for a specified head entity, calculated as the ratio of unique head entities to total head-tail pairs linked by the relation. Conversely, inverse functionality quantifies the tail entity uniqueness for a specified head entity. Formally, these are defined as:

$$\mathcal{F}(r) \coloneqq \frac{|\{e_h|(e_h, r, e_t) \in \mathcal{T}\}}{|\{(e_h, e_t)|(e_h, r, e_t) \in \mathcal{T})\}|}, \quad \mathcal{F}^{-1}(r) \coloneqq \frac{|\{e_t|(e_h, r, e_t) \in \mathcal{T}\}}{|\{(e_h, e_t)|(e_h, r, e_t) \in \mathcal{T})\}|}. \tag{5.4}$$

For instance, suppose a KG contains two triples for the relation $locate\_in$: $(Hawaii, locate\_in, US)$ and $(Miami, locate\_in, US)$. Then $\mathcal{F}(locate\_in) = 1.0$ and $\mathcal{F}^{-1}(locate\_in) = 0.5$. In other words, given $(Miami, locate\_in, ?)$, the answer for the missing tail entity is unique; while given $(?, locate\_in, US)$, there are multiple answers for the missing head entity. Such relational patterns are useful for identifying an entity based on its connections within the intra-graph structure.

Probabilistic reasoning. If two entities are each connected to entities that are aligned across KGs, this increases the likelihood that they should be aligned as well. Based on this heuristic, an entity pair's alignment probability $P(e_h \equiv e'_h)$ can be inferred by aggregating its neighbors' alignment probability via relation functionality:

$$1 - \prod_{\substack{(e_h,r,e_t)\in\mathcal{T}, \\ (e'_h,r',e'_t)\in\mathcal{T}'}} \left(1 - \mathcal{F}^{-1}(r)P(r \subseteq r')P(e_t \equiv e'_t)\right) \times \left(1 - \mathcal{F}^{-1}(r')P(r' \subseteq r)P(e_t \equiv e'_t)\right). \tag{5.5}$$

Here, $P(r \subseteq r')$ denotes the probability of r being a subrelation of r', estimated by alignment probabilities of connected entities:

$$\frac{\sum\left(1 - \prod_{(e'_h,r',e'_t)\in\mathcal{T}'}\left(1 - P(e'_h \equiv e_h)P(e'_t \equiv e_t)\right)\right)}{\sum\left(1 - \prod_{e'_h,e'_t\in\mathcal{E}'}\left(1 - P(e'_h \equiv e_h)P(e'_t \equiv e_t)\right)\right)}. \tag{5.6}$$

These formulations allow for the propagation and updating of alignment probabilities in a manner that is cognizant of relational structures. We employ this technique to design a label refiner below.

Label refiner

Label incompatibility. We exploit the "incompatibility" of labels for label refinement, based on the assumption that correct labels can infer each other, while a false label could be incompatible with its correctly aligned neighbors. We define the *overall*

incompatibility on a label set $\mathcal{L}$ as:

$$\Phi(\mathcal{L}) \coloneqq \sum_{(e_h, e'_h)\in\mathcal{L}} \left(\mathbf{1}_{P(e_h\equiv e'_h)<\max_{e\in\mathcal{E}} P(e,e'_h)} + \mathbf{1}_{P(e_h\equiv e'_h)<\max_{e'\in\mathcal{E}'} P(e_h,e')}\right). \tag{5.7}$$

Here, $\mathbf{1}_{P(e_h\equiv e'_h)<\max_{e\in\mathcal{E}} P(e,e'_h)} = 1$ if e_h is not the top-match for e'_h, otherwise 0. It's important to note that a detected incompatibility doesn't necessarily indicate the false alignment of (e_h, e'_h): it may suggest a misalignment of their neighbors. Given this, the key to label refinement is to jointly optimize the label's overall incompatibility while avoiding accidentally filtering out correct labels.

Objective. To enhance label quality, we propose to refine the pseudo-label set $\mathcal{L}$ by finding a subset $\mathcal{L}^* \subset \mathcal{L}$ that minimizes its overall incompatibility: $\mathcal{L}^* = \text{argmin}_{\mathcal{L}'\subset\mathcal{L}}\, \Phi(\mathcal{L}')$. Noteworthy that a trivial solution for this optimization problem is only preserving a set of isolated labels, such that $\max_{e\in\mathcal{E}} P(e \equiv e'_h) = 0$ and $\max_{e'\in\mathcal{E}'} P(e_h \equiv e') = 0$ for all $(e_h, e'_h) \in \mathcal{L}'$. This trivial solution would lead to the exclusion of most accurate labels, an outcome we aim to avoid. Considering this, we introduce an $l1$ penalty term to penalize the removal of labels, leading to our overall objective:

$$\mathcal{L}^* = \text{argmin}_{\mathcal{L}'\subset\mathcal{L}} \left(\Phi(\mathcal{L}') + \lambda|\mathcal{L} - \mathcal{L}'|\right). \tag{5.8}$$

Here $\lambda > 0$ is a weight coefficient. Solving the above combinatorial problem is intractable as it requires computing $\Phi(\mathcal{L}')$ for each possible set $\mathcal{L}' \subset \mathcal{L}$, which is NP-hard. Below we propose to search for a local-optimal solution by a greedy algorithm powered by probabilistic reasoning.

Greedy search. The algorithm begins by initializing the alignment probability $P(e \equiv e') = \delta_0$ for every pair (e, e') within the set $\mathcal{L}$, where δ_0 is a constant within the range (0,1). It then iteratively performs a search for an optimal label set $\mathcal{L}'$ through a series of voting steps. Each iteration is comprised of two main steps: probabilistic reasoning and label adjustment.

During the probabilistic reasoning step, the alignment probabilities and subrelation probabilities are updated according to Eq. equation (5.5) and Eq. equation (5.6),

respectively. This update process refines our estimates of label confidence based on the latest information. During the label adjustment step, the label set $\mathcal{L}'$ is updated based on these updated probabilities. Labels are appended to $\mathcal{L}'$ if their updated alignment probabilities exceed δ_0, indicative of high confidence in their alignment, supported by their neighbors:

$$\mathcal{L}' \leftarrow \mathcal{L}' \cup \{(e_h, e'_h) \in \mathcal{L} | P(e_h \equiv e'_h) > \delta_0\}. \tag{5.9}$$

Conversely, labels demonstrating structural incompatibilities are excluded from $\mathcal{L}'$:

$$\mathcal{L}' \leftarrow \mathcal{L}' \setminus \left\{(e_h, e'_h) \in \mathcal{L} \mid P(e_h \equiv e'_h) < \max\left(\max_{e \in \mathcal{E}} P(e, e'_h), \max_{e' \in \mathcal{E}'} P(e_h, e')\right)\right\}. \tag{5.10}$$

In this manner, labels are removed if they are incompatible with updated aligned neighbors, ensuring the preservation of only the most confident pairs within $\mathcal{L}'$. To further refine the search process in subsequent iterations, we augment all entity alignment probabilities within $\mathcal{L}'$ to a superior score:

$$P(e \equiv e') \leftarrow \max(P(e \equiv e'), \delta_1) \quad \text{for each } (e, e') \in \mathcal{L}'. \tag{5.11}$$

Here $\delta_1 \in (\delta_0, 1)$ serves as a new threshold, elevating the alignment probabilities of confident pairs to foster a more directed and effective search. After n_{lr} iterations, we get a set of confidently selected labels $\mathcal{L}^*$ that have high compatibility. The detailed algorithm is presented in Algorithm 1, and analyses of parameter efficiency and computational efficiency are provided in Section 5.3.

Below we present the pseudo-code of the greedy algorithm, that incorporates probabilistic reasoning to refine the label set.

5.2.4 Entity alignment

With the refined labels, we train an embedding-based EA model to learn structure-aware representations for each entity. After training, the EA model computes a matching score $m_\theta(e, e')$ for each entity pair (e, e') for evaluation. The selection of

Algorithm 1 The greedy label refinement algorithm

Inputs: The pseudo-label set $\mathcal{L}$

Parameters: The intialization probability $\delta_0 \in (0, 1)$, the threshold $\delta_1 \in (\delta_0, 1)$, probabilistic reasoning iterations n_{lr}

Outputs: The refined pseudo-label set $\mathcal{L}^* \subset \mathcal{L}$

$\mathcal{L}' \leftarrow \varnothing$.

$\forall e \in \mathcal{E}, \forall e' \in \mathcal{E}'$, $P(e \equiv e') \leftarrow 0$

$\forall (e, e') \in \mathcal{L}$, $P(e \equiv e') \leftarrow \delta_0$

$i \leftarrow 0$

while $i < n_{lr}$ **do**

$\forall e_h \in \mathcal{E}, \forall e'_h \in \mathcal{E}'$, $P(e_h \equiv e'_h) \leftarrow 1 - \prod_{\substack{(e_h, r, e_t) \in \mathcal{T}, \\ (e'_h, r', e'_t) \in \mathcal{T}'}} \left(1 - \mathcal{F}^{-1}(r) P(r \subseteq r') P(e_t \equiv e'_t)\right) \times \left(1 - \mathcal{F}^{-1}(r') P(r' \subseteq r) P(e_t \equiv e'_t)\right)$. /* Update entity alignment probabilities.*/

$\forall r \in \mathcal{R}, \forall r' \in \mathcal{R}'$, $P(r \subseteq r') \leftarrow \frac{\sum\left(1 - \prod_{(e'_h, r', e'_t) \in \mathcal{T}'} \left(1 - P(e'_h \equiv e_h) P(e'_t \equiv e_t)\right)\right)}{\sum\left(1 - \prod_{e'_h, e'_t \in \mathcal{E}'} \left(1 - P(e'_h \equiv e_h) P(e'_t \equiv e_t)\right)\right)}$ /* Update subrelation probabilities. */

$\mathcal{L}' \leftarrow \mathcal{L}' \cup \{(e_h, e'_h) \in \mathcal{L} | P(e_h \equiv e'_h) > \delta_0\}$ /* Label adjustment, add confident pairs to the label set.*/

$\mathcal{L}' \leftarrow \mathcal{L}' \setminus \{(e_h, e'_h) \in \mathcal{L} \mid P(e_h \equiv e'_h) < \max\left(\max_{e \in \mathcal{E}} P(e, e'_h), \max_{e' \in \mathcal{E}'} P(e_h, e')\right)\}$ /* Label adjustment, remove less confident pairs from the label set. */

$P(e \equiv e') \leftarrow \max\left(P(e \equiv e'), \delta_1\right)$ for each $(e, e') \in \mathcal{L}'$ /* Elevate alignment probability of confident pairs. */

end while

$\mathcal{L}^* \leftarrow \mathcal{L}' \cup \{(e, e') | P(e \equiv e') > \delta_1\}$ /* Augment the refined label set with confident pairs.*/

Return $\mathcal{L}^*$

=0

the base EA model is flexible, tailored to the task requirements. We chose a recently proposed GCN-based model, Dual-AMN [31], for its effectiveness and efficiency.

Feedback from the base EA model is crucial for dynamic update of the active selection policy. To generate effective feedback, we infer high-confidence pairs (e, e') with the

trained EA model, by selecting the pairs that both entities rank top for each other. These pairs are injected into the probabilistic reasoning system. Similar to the label refinement process, this system initializes with an alignment probability of δ_0 for these pairs and updates the estimation of alignment and subrelation probabilities using Eq. equation (5.5) and Eq. equation (5.6). The updated probabilities are used to construct the uncertainty terms (i.e., U_r and U_n) to inform the active selection policy in subsequent iterations, thereby optimizing the budget utility and improving final performance continuously.

5.3 Efficient implementation of label refiner

Parameter-efficient probabilistic reasoning. The total number of alignment probabilities for all entity pairs is $|\mathcal{E}||\mathcal{E}'|$, resulting in a large parameter size when the KGs involved are extensive. We enhance memory efficiency by adopting a lazy inference strategy in probabilistic reasoning. This strategy involves only saving the alignment probabilities of the most probable alignments:

$$\left\{ P(e_h, e'_h), |, e_h \in \mathcal{E}, e'_h \in \mathcal{E}', P(e_h \equiv e'_h) = \max\left(\max_{e \in \mathcal{E}} P(e, e'_h), \max_{e' \in \mathcal{E}'} P(e_h, e') \right) \right\}, \tag{5.12}$$

Probabilities of other entity pairs can be inferred from these saved alignment probabilities using Eq. equation (5.5) when necessary. In this way, parameter complexity is reduced to $O(\max(|\mathcal{E}| + |\mathcal{E}'|))$.

Computation efficiency of probabilistic reasoning. Probabilistic reasoning is executed iteratively, with each iteration updating the alignment probabilities of all entities and relations following Eq. equation (5.5) and Eq. equation (5.6). We detail the analysis of these two update phases in the following separately.

Since we adopt a lazy inference strategy, the update of entity alignment probabilities involves updating the set of most probable alignments and associated probabilities as

shown in Eq. equation (5.12). This update requires the estimation of all $P(e, e'_h), e \in \mathcal{E}$ and all $P(e_h, e'), e' \in \mathcal{E}'$, for each current pair (e_h, e'_h) to determine if this pair needs an update. Consequently, the computational complexity of this process is proportional to the size of this set, which is $O(|\mathcal{E}|)$. The estimated scores $P(e, e'_h), e \in \mathcal{E}$ and $P(e_h, e'), e' \in \mathcal{E}'$ can be precomputed in advance and reused for all pairs (e_h, e'_h), leading to a computation complexity of $O(|\mathcal{E}||\mathcal{E}'|)$. Thus, the overall computational complexity for updating entity alignment probabilities is $O(|\mathcal{E}||\mathcal{E}'| + |\mathcal{E}|) = O(|\mathcal{E}||\mathcal{E}'|)$.

The update process for sub-relation probabilities involves updating all $P(r \subset r')$ for $r \in \mathcal{R}$ and $r' \in \mathcal{R}'$, resulting in a complexity of $O(|\mathcal{R}||\mathcal{R}'|)$. The estimation of Eq. equation (5.6) utilizes the probabilities of the most probable alignments from Eq. equation (5.12). Notably, most relation pairs (r, r') do not have aligned head entities (e_h, e'_h) or aligned tail entities (e_t, e'_t), thus most relation pairs can be excluded for efficient computation by exploiting this sparsity heuristic, reducing the computations by orders.

It is worth noting that these computations can be further accelerated through parallelization, as their execution solely depends on the results from the previous iteration.

5.4 Experiments

In this section, we conduct experiments to evaluate the effectiveness of our framework. We begin by introducing the experimental settings. Then, we present experiments to answer the following research questions: **RQ1.** How effective is the overall framework? **RQ2.** What is the impact of the choice of LLM on the cost and performance of LLM4EA? **RQ3.** What is the effect of the label refiner? **RQ4.** What is the impact of active selection? **RQ5.** How is the runtime-performance trade-off managed?

5.4.1 Experimental setting

Datasets and LLM. In this study, we use the widely-adopted OpenEA dataset [48], including two monolingual datasets (D-W-15K and D-Y-15K) and two cross-lingual datasets (EN-DE-15K and EN-FR-15K). OpenEA comes in two versions: "V1" the normal version, and "V2" the dense version. We have chosen "V2" because it more closely resembles existing KGs. Datasets are preprocessed before used, as detailed in Section 5.4.1. The LLM version in this experiment is GPT-3.5 (gpt-3.5-turbo-0125) and GPT-4 (gpt-4-turbo-2024-04-09). By default, the overall query budget is $\mathcal{B} = 0.1|\mathcal{E}|$.

Baselines. Baseline models include three GCN-based models — GCNAlign [55], RDGCN [57], Dual-AMN [31], and three translation-based models — IMUSE [19], AlignE, BootEA [47], Here, BootEA is a variant of AlignE that adopts a bootstrapping strategy, equipped with a label calibration component for improving the accuracy of bootstrapped labels. Baseline models are directly trained on the pseudo-labels generated by the LLM annotator, without label refinement or active selection. Every experiment is repeated three times to report statistics.

Setup of LLM4EA. We employ GPT-3.5 as the default LLM due to its cost efficiency. Other parameters are $n = 3$, n_{lr}, $k = 20$, $\delta_0 = 0.5$, $\delta_1 = 0.9$.

Hardware and software configurations

Our experiments were conducted on a server equipped with six NVIDIA GeForce RTX 3090 GPUs, 48 Intel(R) Xeon(R) Silver 4214R CPUs, and 376GB of host memory. The models were implemented using TensorFlow, NumPy, and SciPy. It was observed that the software version significantly affects hardware-software compatibility. Specifically, the original implementation of Dual-AMN was based on TensorFlow 1.14.0, which is not compatible with newer GPUs such as the NVIDIA GeForce RTX 3090. Therefore,

we updated the code to be compatible with TensorFlow 2.7.0, enabling the model to leverage GPU acceleration effectively. The details of the software packages used in our experiments are listed in Table 5.1.

Table 5.1: Package configurations of our experiments.

Package	tqdm	numpy	scipy	tensorflow	keras	openai
Version	4.66.2	1.24.4	1.10.1	2.7.0	2.7.0	1.30.1

Dataset statistics and preprocessing

Table 5.2: Data statistics of used OpenEA dataset.

Datasets	KG	#Rel.	#Rel tr.	#Attr.	#Attr tr.	#Named Ent.	#Targets in top-k
EN-FR	English (EN)	193	96,318	189	66,899	15,000	13,550
	French (FR)	166	80,112	221	68,779	15,000	
EN-DE	English (EN)	169	84,867	171	81,988	15,000	13,330
	German (DE)	96	92,632	116	186,335	15,000	
D-W	DBpedia (DB)	167	73,983	175	66,813	15,000	12,910
	Wikidata (WD)	121	83,365	457	175,686	13,458	
D-Y	DBpedia (DB)	72	68,063	90	65,100	15,000	15,000
	Yago (YG)	21	60,970	20	131,151	15,000	

In our experiments, we utilized the OpenEA dataset version 1.1 (V2), specifically the 15K set. The statistics are detailed in Table 5.2. It's important to highlight that the destination dataset for D-W-15K, originating from Wikidata, contains only entity IDs, lacking explicit names. These IDs, devoid of semantic content, are not inherently meaningful to a language model. To rectify this, we processed the dataset using the 'wikidatawiki-20160801-abstract.xml' dump file from Wikidata. This file provided the raw data necessary for constructing the D-W-15K dataset, enabling us to extract meaningful entity names.

We shown the quantity of entities with names (after name extraction) for each KG in the 'Named entities' column in Table 5.2.

In the counterpart filtering phase, we selected the top-k (where $k = 20$ for our experiments) most similar candidates. The 'Target in top-k' column of Table 5.2 shows the number of target entities included in this selection.

5.4.2 Results

Comprehensive evaluation of entity alignment performance

Table 5.3: Evaluation of entity alignment performance, measured by Hit@K for $K \in \{1, 10\}$, and Mean Reciprocal Rank (MRR), presented in %. Experiment statistics are computed over three trials.

	EN-FR-15K			**EN-DE-15K**			**D-W-15K**			**D-Y-15K**		
	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR
Group1. Entity Alignment with GPT-3.5.												
IMUSE	50.0±0.1	72.6±0.8	57.5±0.4	51.6±4.7	75.9±3.9	60.5±4.5	6.0±0.2	14.6±2.5	9.0±1.0	54.4±2.5	78.9±1.1	63.2±2.0
AlignE	6.6±0.3	24.5±0.5	12.6±0.5	6.2±0.3	18.4±1.0	10.4±0.5	8.0±0.9	24.0±2.7	13.3±1.4	50.1±2.0	76.6±1.4	59.2±1.8
BootEA	44.8±1.1	71.9±1.2	54.2±1.2	68.1±0.2	85.4±0.3	74.3±0.2	60.8±0.2	79.3±0.1	67.4±0.2	87.8±0.1	96.7±0.1	91.2±0.1
GCNAlign	17.4±0.3	43.2±0.4	25.9±0.3	22.2±0.2	46.2±1.1	30.3±0.3	16.9±0.1	39.3±0.3	24.3±0.1	45.3±0.4	68.3±0.6	53.3±0.5
RDGCN	69.3±0.3	82.5±0.3	74.3±0.3	73.3±4.3	84.6±2.6	77.4±3.7	79.2±0.7	89.7±0.5	83.2±0.6	82.6±3.7	91.9±1.3	86.1±2.7
Dual-AMN	51.9±0.3	79.6±0.9	61.6±0.5	70.5±0.7	91.1±0.3	78.9±0.6	62.0±0.1	86.8±0.1	71.9±0.1	85.8±0.3	98.4±0.0	91.4±0.1
LLM4EA	**74.2±0.3**	**92.9±0.4**	**81.0±0.3**	**89.1±0.5**	**97.8±0.1**	**92.6±0.3**	**87.5±0.3**	**96.7±0.1**	**90.9±0.2**	**97.7±0.0**	**99.5±0.0**	**98.3±0.0**
Group2. Entity Alignment with GPT-4.												
IMUSE	52.7±0.9	74.9±1.0	59.8±0.9	59.6±2.6	81.8±1.5	67.9±2.1	21.6±6.1	50.0±10.0	31.1±7.4	86.6±0.5	94.2±0.1	89.2±0.4
AlignE	30.8±2.4	69.1±2.5	43.1±2.5	46.4±5.2	76.5±3.8	56.6±4.8	36.1±3.7	67.8±3.6	46.7±3.7	86.4±0.9	97.0±0.3	90.2±0.6
BootEA	58.2±0.3	83.7±0.3	67.0±0.3	80.5±0.4	92.6±0.2	84.8±0.3	71.6±0.2	88.3±0.2	77.6±0.2	95.0±0.1	98.6±0.0	96.3±0.1
GCNAlign	30.6±0.0	65.3±0.3	42.1±0.2	41.9±0.4	68.6±0.5	51.2±0.4	31.3±0.3	61.6±0.1	41.4±0.2	82.6±0.2	94.9±0.2	87.2±0.1
RDGCN	72.1±0.2	84.5±0.1	76.7±0.2	74.1±1.1	85.1±0.7	78.0±1.0	82.5±1.1	91.4±0.7	85.9±1.0	85.4±0.9	93.2±0.4	88.3±0.8
Dual-AMN	76.7±0.1	94.9±0.3	83.6±0.2	90.7±0.1	97.9±0.2	93.6±0.1	81.5±0.1	94.9±0.2	86.7±0.1	97.5±0.0	99.3±0.1	98.1±0.0
LLM4EA	**80.2±0.3**	**96.0±0.2**	**86.0±0.2**	**93.1±0.5**	**98.7±0.2**	**95.3±0.3**	**89.8±0.3**	**97.9±0.2**	**92.9±0.3**	**97.9±0.1**	**99.6±0.0**	**98.5±0.1**

To answer **RQ1** and **RQ2**, we conducted two groups of experiments on OpenEA datasets, using GPT-3.5 and GPT-4 as the annotator, respectively. Results are presented in Table 5.3. We also investigated the performance-cost comparison between

the GPT-3.5 annotator and the GPT-4 annotator, illustrated in Figure 5.2. To control the randomness introduced by the LLMs, each experiment was repeated three times to report mean and standard deviation. These results lead to several key observations:

First, LLM4EA surpasses all baseline EA models, which are directly trained on the pseudo-labels, by a large margin. This can be attributed to 1) our label refiner's capability in filtering out false labels, reducing noise during training and enabling more accurate optimization towards the ground true objective; 2) our active selection component's ability to smartly identify important entities to annotate, which takes full advantage of the fixed query budget.

Second, using the GPT-4 results in higher performance than using the GPT-3.5 as the annotator. This observation conforms to the fact that GPT-4 is a more advanced LLM with higher reasoning capacity and stronger semantic analysis, resulting in more precise annotation results and higher recall, thus providing more labels of high quality. We also observe that translation-based models (e.g., AlignE) are sensitive to noisy labels under GPT-3.5, while state-of-the-art GCN-based models (e.g., RDGCN and Dual-AMN) are more robust. BootEA also demonstrates superior performance and robustness, attributed to its bootstrapping technique and enhanced by its capability in calibrating bootstrapped labels. However, its label calibration is only applied to the bootstrapped labels, so it still suffers from the false training labels. Our proposed LLM4EA, on the other hand, refines the label accuracy before training the EA model, thus ensuring more accurate training.

Finally, LLM4EA is noise adaptive, enabling cost-efficient entity alignment. To further investigate the effect of the choice of LLM, we examined the performance-cost comparison between GPT-3.5 and GPT-4 as the annotator. We illustrate MRR in Figure 5.2 (detailed results are available in Table 5.4). The results show that, by increasing the query budgets (measured by the number of tokens) for GPT-3.5, the performance gradually increases. When the budget is 2× that of GPT-4, the performance is comparable to or exceeds the performance of using GPT-4 as the

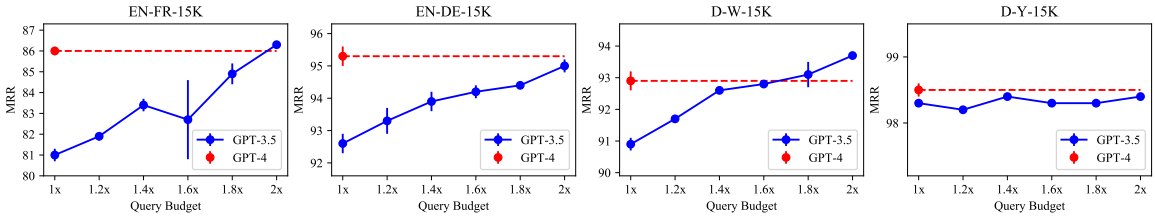

Figure 5.2: Performance-cost comparison between GPT-3.5 and GPT-4 as the annotator, evaluated by MRR. We increase the budget for GPT-3.5 to evaluate its performance. [$n\times$] denotes using $n\times$ of the default query budget. Each experiment is repeated three times to show mean and standard deviation.

Table 5.4: Performance-cost comparison between GPT-3.5 and GPT-4 as annotator. We increase the budget for GPT-3.5 to evaluate its performance. [$n\times$] denotes using $n\times$ of the default query budget.

	EN-FR-15K			**EN-DE-15K**			**D-W-15K**			**D-Y-15K**		
	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR
GPT-3.5	74.2±0.3	92.9±0.4	81.0±0.3	89.1±0.5	97.8±0.1	92.6±0.3	87.5±0.3	96.7±0.1	90.9±0.2	97.7±0.0	99.5±0.0	98.3±0.0
GPT-3.5 (1.2×)	75.4±0.4	93.2±0.4	81.9±0.2	90.2±0.6	98.0±0.0	93.3±0.4	88.4±0.1	97.1±0.2	91.7±0.1	97.6±0.0	99.3±0.1	98.2±0.0
GPT-3.5 (1.4×)	77.2±0.2	94.5±0.5	83.4±0.3	91.0±0.4	98.1±0.1	93.9±0.3	89.2±0.2	97.9±0.0	92.6±0.1	97.7±0.0	99.5±0.0	98.4±0.0
GPT-3.5 (1.6×)	76.3±2.4	94.1±0.7	82.7±1.9	91.4±0.2	98.4±0.0	94.2±0.2	89.6±0.0	97.9±0.2	92.8±0.1	97.7±0.0	99.4±0.0	98.3±0.0
GPT-3.5 (1.8×)	78.8±0.5	95.2±0.4	84.9±0.5	91.9±0.1	98.1±0.0	94.4±0.1	90.1±0.5	97.9±0.2	93.1±0.4	97.7±0.0	99.5±0.1	98.3±0.0
GPT-3.5 (2×)	**80.6±0.2**	95.9±0.1	**86.3±0.1**	92.7±0.2	98.5±0.1	95.0±0.2	**90.7±0.2**	**98.5±0.1**	**93.7±0.1**	97.8±0.0	99.5±0.0	98.4±0.0
GPT-4	80.2±0.3	**96.0±0.2**	86.0±0.2	**93.1±0.5**	**98.7±0.2**	**95.3±0.3**	89.8±0.3	97.9±0.2	92.9±0.3	**97.9±0.1**	**99.6±0.0**	**98.5±0.1**

annotator. According to the pricing scheme of OpenAI, the input/output cost for 1 million tokens for GPT-3.5 and GPT-4 is \$0.50/\$1.5 and \$10/\$30, respectively. This means that our noise-adaptive framework enables cost-efficient entity alignment with less advanced LLMs at 10× less actual cost than using more advanced LLMs, simply by increasing the token budget for the less advanced LLMs.

Effect of the label refiner

To answer **RQ3**, we first analyze the evolution of the True Positive Rate (TPR) and the recall rate of the refined labels. Specifically, at each label refinement iteration,

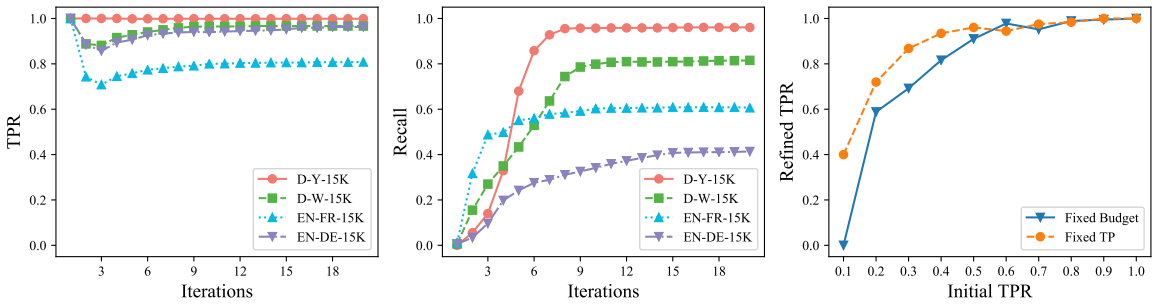

Figure 5.3: Analysis of the Label Refinement. We illustrate the evolution of the true positive rate (TPR) (left) and recall (middle) for refined labels across four datasets. Furthermore, we assess the robustness of the label refinement process by examining the TPR of refined labels against varying initial TPRs within the D-W-15K dataset (right), with initial pseudo-labels synthesized at different TPR levels.

the TPR is calculated as $\frac{|\mathcal{A} \cap \mathcal{L}'|}{|\mathcal{L}'|}$, and the recall is calculated as $\frac{|\mathcal{A} \cap \mathcal{L}'|}{|\mathcal{A} \cap \mathcal{L}|}$. The left and middle subfigures of Figure 5.3 demonstrate how **our label refiner progressively discovers accurate labels and optimizes the TPR.** Initially, the TPR of the refined label set is high (approximately 1.0), then it decreases by a certain percentage, and eventually increases again to a high TPR. We attribute this pattern to: 1) the most confident labels being discovered in the earliest iterations, which are obvious alignments with many connected alignments; 2) as the algorithm progresses, some false pseudo-labels being erroneously added to the label set $\mathcal{L}'$; 3) as the label refinement continues, $\mathcal{L}'$ is adjusted and the false pseudo-labels are replaced with the correct labels inferred by the updated probability as in Eq. equation (5.10).

Furthermore, we assess the robustness of our label refiner, as depicted in the right subfigure of Figure 5.3. We synthesize noisy labels and evaluate the output TPR in relation to varying input TPR levels, using two experimental schemes: *fixed budget*, where the budget remains constant at $0.1|\mathcal{E}|$ while the TPR changes, and *fixed TP*, where the number of true positives is fixed but the TPR and corresponding budgets are adjusted. The results demonstrate that **the label refiner consistently elevates the TPR to over** 0.9**, even when the initial TPR is around** 0.5**, showcasing its high**

robustness to noisy pseudo-labels. This result also reveals why our framework demonstrates robust performance with the less advanced GPT-3.5 annotator.

Ablation study

Table 5.5: Ablation study overview. The table presents the performance of the LLM4EA (Ours) with various modifications. **Group 1**: removing the label refiner (w/o LR) and the active selection component (w/o Act); **Group 2**: replacing the active selection technique with relational uncertainty (-ru), neighbor uncertainty (-nu), degree (-degree), and functionality sum (-funcSum).

	EN-FR-15K			**EN-DE-15K**			**D-W-15K**			**D-Y-15K**		
	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR
Ours	74.2±0.3	92.9±0.4	81.0±0.3	**89.1±0.5**	**97.8±0.1**	**92.6±0.3**	87.5±0.3	96.7±0.1	90.9±0.2	**97.7±0.0**	**99.5±0.0**	**98.3±0.0**
w/o LR	51.6±1.0	80.2±0.7	61.9±0.8	74.4±1.7	94.2±0.6	82.6±1.3	39.2±1.4	75.7±0.7	52.9±1.2	85.3±1.0	99.2±0.1	91.5±0.5
w/o Act	68.1±2.1	88.4±1.7	75.4±2.0	78.4±0.8	93.9±0.3	84.6±0.6	82.8±0.7	92.5±0.6	86.3±0.6	97.5±0.1	99.2±0.3	98.1±0.1
Ours-ru	70.8±1.0	91.2±0.4	78.2±0.8	83.9±0.3	97.7±0.2	89.6±0.2	**88.7±0.6**	**97.4±0.3**	**92.1±0.5**	**97.7±0.0**	99.4±0.1	**98.3±0.0**
Ours-nu	**74.5±0.7**	**93.1±0.5**	**81.2±0.6**	88.8±0.2	96.7±0.3	91.8±0.3	85.1±0.5	95.2±0.5	88.9±0.5	97.6±0.1	99.4±0.0	98.2±0.0
Ours-degree	73.6±2.6	92.5±0.8	80.4±2.0	88.4±0.1	96.6±0.2	91.5±0.2	80.1±3.7	90.9±2.2	84.0±3.2	97.2±0.2	99.0±0.1	97.9±0.1
Ours-funcSum	59.5±0.6	78.8±0.6	66.3±0.6	81.2±0.5	96.0±0.3	87.1±0.4	83.9±0.9	93.1±1.1	87.3±1.0	97.5±0.1	99.4±0.1	98.1±0.1

Ablation studies detailed in Table 5.5 answer **RQ4** and reveal several key insights: **1) Necessity of the Label Refiner for Effective Active Selection:** The performance of "w/o LR", which lacks a label refiner, is inferior not only to other model variants but also to the base model, Dual-AMN. This underscores that active selection depends crucially on reliable feedback, which is compromised when the label refiner is absent; **2) Contribution of Relation and Neighbor Uncertainty in Active Selection:** Incorporating both relation and neighbor uncertainties significantly enhances the utility of the budget. Methods like "Ours-degree" and "Ours-funcSum" focus only on their connections to neighbors and ignore the uncertainty of neighbors. In contrast, "Ours-ru" and "Ours-nu", which take these uncertainties into account, exhibit superior performance. This underscores the importance of considering neighbor uncertainty for effective active selection; **3) Robust Active Selection through Combined**

Metrics: Our active selection approach integrates both relation uncertainty and neighbor uncertainty to enable robust active selection. By employing rank aggregation, it prioritizes entities that are deemed significant by both metrics, ensuring a more effective and nuanced selection process.

Pareto frontier of runtime overhead against performance.

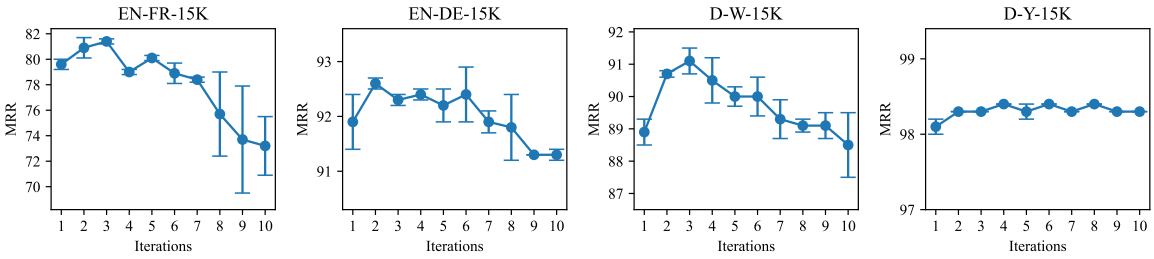

Figure 5.4: Performance of entity alignment across four datasets with varying active sampling iterations, under a fixed query budget.

We answer **RQ5.** in this experiment. The runtime overhead is directly proportional to the number of active selection iterations n, since each iteration involves a subsequent label refinement process. To explore the runtime-performance trade-off, we examine the Pareto frontier of runtime versus performance. We conduct entity alignment experiments with a fixed query budget, varying the number of active selection iterations. The results of these experiments are illustrated in Figure 5.4.

The results indicate that performance initially increases as the number of iterations rises from 1 to around 3, but further increases beyond this point lead to a decline. This pattern can be attributed to two main factors: (1) More iterations allow for extensive learning from feedback during the active selection phase. (2) However, when iterations are excessively high, the number of generated pseudo-labels per iteration becomes small, leading to isolated pseudo-labels that undermine the label refinement process. This occurs because label refinement heavily relies on the compatibility of the local structure, causing correct labels to be identified less frequently and

making the feedback for active selection less informative. These findings suggest that **an optimal balance between runtime efficiency and performance can be achieved without excessive trade-offs**, indicating a specific threshold for iterations beyond which no further performance gains are observed.

Performance comparison against rule-based models

Table 5.6: Performance comparison against rule-based models, evaluated by precision, recall, and f1-score.

	EN-FR-15K			**EN-DE-15K**			**D-W-15K**			**D-Y-15K**		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Emb-Match	**90.1**	4.4	8.4	69.3	3.1	6.0	80.8	3.4	6.6	**100**	1.0	2.0
Str-Match	84.8	**69.8**	**76.6**	**92.3**	71.4	80.5	**96.2**	57.9	72.3	76.9	**100**	86.9
PARIS	58.3±0.5	26.5±0.3	36.5±0.4	90.8±0.3	50.7±0.3	65.0±0.3	92.4±0.4	70.2±0.2	79.8±0.2	99.1±0.1	96.7±0.1	97.9±0.1
LLM4EA	68.6±0.3	53.1±0.2	59.8±0.2	90.5±0.3	**82.4±0.4**	**86.2±0.4**	90.7±0.4	**81.6±0.5**	**85.9±0.5**	98.9±0.0	97.6±0.1	**98.3±0.0**

In this section, we compare the LLM4EA model with several rule-based models, including two lexical matching-based approaches: Emb-Match and Str-Match. Emb-Match uses cosine similarities between word embeddings to identify aligned pairs, employing the fasttext-wiki-news-subwords-300 model, which can handle unseen words due to its subword capabilities. Str-Match utilizes the Levenshtein Distance to calculate similarity scores. Additionally, the probabilistic reasoning model PARIS performs entity alignment by relying on probabilistic methods.

For the lexical matching-based models, we compute the similarity and evaluate the confident entity pairs, specifically targeting those whose normalized similarity scores exceed 0.8. The same word embedding model used for Emb-Match is 'fasttext-wiki-news-subwords-300', notable for its ability to process unseen words as a subword model. For PARIS, we assess all inferred aligned pairs by setting the threshold to zero. The entity alignment (EA) model of LLM4EA generates a ranked score list, which is not directly comparable with these rule-based models. To facilitate comparison, we

use the trained EA model to generate confident pairs, ensuring that each entity is the top-ranked candidate for its counterpart. These pairs are then processed through our label refiner, and the refined pairs are evaluated. Experiments for PARIS and LLM4EA are repeated three times to ensure statistical reliability; for Emb-Match and Str-Match, experiments are performed once as these algorithms are deterministic. The results are presented in Table 5.6.

The findings reveal that: 1) Emb-Match demonstrates low recall due to its inability to fully grasp the semantics of many unseen words, despite generating embeddings for them; 2) Str-Match performs well on these datasets because most names are identical, thus enhancing the precision of its string matching algorithm; 3) PARIS delivers precise inference results by managing noisy data through probabilistic reasoning, although it shows lower recall than LLM4EA; 4) LLM4EA, benefiting from its active selection and label refinement techniques, demonstrates robust and precise performance across all datasets.

Chapter 6

Conlusion and future work

6.1 Conlusion

In this thesis, we present algorithms and applications for Knowledge Graph (KG) reasoning. We start by introducing neural, symbolic, and neuro-symbolic algorithms for reasoning within a single KG, and propose a differentiable neuro-symbolic KG reasoning framework, DiffLogic, for the KG completion task. We then extend the topic to reasoning algorithms that exploit inter-KG structures and propose a probabilistic reasoning system for entity alignment between two KGs. This probabilistic reasoning not only infers new alignments but also provides robust label refinement capabilities that continuously improve the accuracy of noisy labels. Based on this reasoning system, we propose a novel framework, LLM4EA, that enables label-free entity alignment with noisy annotations from large language models. We also employ feedback from the reasoning system to inform an active selection component, continuously improving the utility of the fixed query budget to the LLMs. Finally, we introduce an acceleration algorithm that employs randomized sparse computation to accelerate graph learning, a core step of state-of-art neural reasoning methods.

6.2 Future Research Directions

In this section, we outline potential avenues for future research that build upon the findings of this thesis, particularly focusing on (1) KG reasoning with quantified uncertainty, and (2) integration of Large Language Models (LLMs) and Knowledge Graphs (KGs).

6.2.1 Conformalized KG reasoning

Existing KG reasoning methods only output a ranked list as the prediction without predicting true/false labels and associated confidence scores. For instance, given a head-relation pair $(h, r, ?)$, existing reasoning models perform the KG completion task by generating a ranked list of matching scores for each possible candidate tail entity but do not produce a set of the most confident triplets each associated with a confidence score. In real-world applications, the ranked score list does not provide trustworthy predictions for downstream tasks. Additionally, annotating these candidate tails is more labor-intensive than labeling a set of high-confidence triplets. To address this, we propose extending conformal prediction—a statistical framework that predicts trustworthy outputs with quantized uncertainty scores—to KG reasoning. This framework can incorporate any off-the-shelf reasoning models and is capable of generating a set of confident triplets with a statistically guaranteed true-positive rate, even if the chosen model does not perform well. The challenge in this task lies in conformal prediction's requirement for the exchangeability of the validation set and the test set, while KG reasoning is intrinsically transductive, and the existence of these triplets depends on other triplets through the graph structure. The core of this work will address the exchangeability requirement and formulate an effective uncertainty score for each prediction.

6.2.2 LLM for Knowledge Graph Construction

Future work will explore the development of advanced LLMs specifically designed for the automatic construction of Knowledge Graphs. This involves enhancing the capabilities of LLMs to extract, organize, and represent knowledge from unstructured data sources, thereby facilitating the creation of comprehensive and dynamic KGs. Investigating the effectiveness of various LLM architectures in this context will be a key focus.

6.2.3 Knowledge Graph for LLM: Graph Retrieval Augmented Generation

Another promising direction is the utilization of KGs to augment LLMs in generating more contextually relevant and accurate outputs. This can be approached in two main ways:

- **KG as Knowledge Carrier**: Future research will examine how KGs can serve as a repository of structured knowledge that LLMs can reference during the generation process. This integration aims to improve the factual accuracy and relevance of the generated content.
- **KG as Index Graph**: Additionally, we will investigate the potential of KGs as index graphs to enhance the retrieval of information. By leveraging the relationships and entities within KGs, LLMs can be guided to access pertinent information more efficiently, thereby improving the overall quality of generated responses.

By pursuing these avenues, we aim to bridge the gap between LLMs and KGs, ultimately leading to more intelligent and context-aware systems capable of handling complex information retrieval and generation tasks.

6.2.4 Building Upon Our Current Work

The current work, including DiffLogic, NeuSymEA, and LLM4EA, lays a strong foundation for future research in KG reasoning and entity alignment by addressing key challenges in scalability, robustness, and label-free learning. Here's how our work connects with the proposed future directions:

- **Foundation for Conformalized KG Reasoning**: Our work in neuro-symbolic reasoning, particularly DiffLogic, provides a framework for efficiently combining symbolic logic with neural networks, making it suitable for integrating conformal prediction methods. The differentiable neuro-symbolic approach allows for scalable reasoning, which can be extended to produce predictions with quantized uncertainty scores. DiffLogic's ability to reason over large-scale KGs with learned rules will be a valuable asset when incorporating uncertainty quantification into predictions, making them more reliable for downstream applications.
- **NeuSymEA as a Precursor to Robust, Adaptive Models**: NeuSymEA's robustness in low-resource settings, achieved through its integration of symbolic reasoning and neural embeddings, sets the stage for adaptive models that can operate in a wide variety of contexts. Its variational EM framework can be further developed to accommodate uncertainty and probabilistic outputs, making it a natural fit for future work in KG reasoning with uncertainty scores. Additionally, its scalable and interpretable alignment framework is well-suited for advancing LLM4EA and exploring hybrid LLM-KG architectures.
- **LLM4EA and Knowledge Integration**: LLM4EA's approach to label-free entity alignment via large language models offers a direct path for exploring deeper integration between LLMs and KGs. The model's capability to handle noisy annotations and refine labels using probabilistic reasoning can serve as a starting point for developing advanced LLMs that interact with KGs to automatically construct, update, and retrieve knowledge. Furthermore, LLM4EA's

active learning mechanism aligns with the goals of using KGs as a structured knowledge base for LLMs, enhancing both the accuracy of generated content and the relevance of retrieved information.

In summary, our current work provides essential methodological and conceptual tools for the proposed future directions. By leveraging the advancements made in neuro-symbolic reasoning, robust entity alignment, and label-free learning, we can expand the scope of KG reasoning and LLM-KG integration, leading to more scalable, interpretable, and context-aware AI systems.

Appendix A

Mathematical Proofs

A.1 Notations of DiffLogic

Table A.1: Notations.

Notation	Description
$\mathcal{K}$	The knowledge base
$\mathcal{E}$	The entity set
$\mathcal{R}$	The relation set
$\mathcal{O}, \mathcal{H}$	The set of observed and unobserved facts
$\boldsymbol{x}, \boldsymbol{y}$	The assignments of $\mathcal{O}$ and $\mathcal{H}$, respectively
$\{F_q, W_q\}_{q=1}^m$	The set of logic rules and attached weights
$\mathcal{I}_q^-, \mathcal{I}_q^+$	The index set of premise atoms and conclusion atoms of rule F_q, respectively
$A, B, ...$	Variables in logic rules
$\{G_q^{(j)}, j \in t_q\}$	All ground formulas created by the q_{th} logic rule
$\Phi_q(\boldsymbol{y}, \boldsymbol{x})$	The sum of potentials of all ground formulas of F_q
θ	The embedding parameters

A.2 Mathematical analysis of DiffLogic

A.2.1 Derivation of rule weight gradient

Given

$$P^*_{\boldsymbol{w}}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{i=1}^{n} P^*\left(y_i \mid \mathrm{MB}\left(y_i\right), \boldsymbol{x}\right) = \prod_{i=1}^{n} \frac{\exp\left[-f^i_{\boldsymbol{w}}\left(y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x}\right)\right]}{Z_i(\boldsymbol{W}, y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x})},$$
$$Z_i(\boldsymbol{W}, y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x}) = \int_{y_i} \exp\left[-f^i_{\boldsymbol{w}}\left(y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x}\right)\right], \quad f^i_{\boldsymbol{w}} = \sum_{q=1}^{m} W_q \sum_{j=1}^{n_q} \mathbf{1}_{\{y_i \to G_q^{(j)}\}} d(G_q^{(j)}),$$

we have

$$\frac{\partial \log P^*(\boldsymbol{y} \mid \boldsymbol{x})}{\partial W_q} = \sum_{i=1}^{n} \frac{\partial \log P^*(y_i \mid \mathrm{MB}\left(y_i\right), \boldsymbol{x})}{\partial W_q}. \tag{A.1}$$

The partial derivative in the left side of Eq. equation (A.1) is a summation of n terms, each term represents the partial derivatives of the pseudo-log-likelihood for each y_i, conditioned on its Markov blankets. Each term can be further simplified as follows:

$$\begin{aligned}
&\frac{\partial \log P^*(y_i \mid \mathrm{MB}\left(y_i\right), \boldsymbol{x})}{\partial W_q} \\
=&\frac{\partial \left\{-f^i_{\boldsymbol{w}}\left(y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x}\right) - \log Z_i(\boldsymbol{W}, y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x})\right\}}{\partial W_q} \\
=&\frac{\partial \left\{-f^i_{\boldsymbol{w}}\left(y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x}\right) - \log \int_{y_i} \exp\left[-f^i_{\boldsymbol{w}}\left(y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x}\right)\right]\right\}}{\partial W_q}.
\end{aligned}$$

Here, we can easily get

$$\frac{\partial f^i_{\boldsymbol{w}}\left(y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x}\right)}{\partial W_q} = \sum_{j} \mathbf{1}_{\{y_i \to G_q^{(j)}\}} d(G_q^{(j)}). \tag{A.2}$$

To make the writing concise, we replace the right term of Eq. equation (A.2) with the

following notation:

$$\Psi_{q,MB(i)} = \sum_j \mathbf{1}_{\{y_i \to G_q^{(j)}\}} d(G_q^{(j)}).$$

In this way, we can deduce that:

$$\begin{aligned}
&\frac{\partial \log P^*(y_i \mid \mathrm{MB}\,(y_i), \boldsymbol{x})}{\partial W_q} \\
=& -\Psi_{q,MB(i)} - \frac{1}{Z_i(\boldsymbol{W}, y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x})} \frac{\partial \int_{y_i} \exp\left[-f_{\boldsymbol{w}}^i\left(y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x}\right)\right]}{\partial W_q}.
\end{aligned} \tag{A.3}$$

The partial derivative and the integration in Eq. equation (A.3) can be swapped using Lebesgue's dominated convergence theorem, the Eq. equation (A.3) thus becomes:

$$\begin{aligned}
&\frac{\partial \log P^*(y_i \mid \mathrm{MB}\,(y_i), \boldsymbol{x})}{\partial W_q} \\
=& -\Psi_{q,MB(i)} - \frac{1}{Z_i(\boldsymbol{W}, y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x})} \int_{y_i} \frac{\partial \exp\left[-f_{\boldsymbol{w}}^i\left(y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x}\right)\right]}{\partial W_q} \\
=& -\Psi_{q,MB(i)} + \int_{y_i} \frac{\exp\left[-f_{\boldsymbol{w}}^i\left(y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x}\right)\right]}{Z_i(\boldsymbol{W}, y_i \cup \boldsymbol{y}_{\setminus i}, \boldsymbol{x})} \Psi_{q,MB(i)} \\
=& -\Psi_{q,MB(i)} + \int_{y_i} P^*\left(y_i \mid \mathrm{MB}\,(y_i), \boldsymbol{x}\right) \Psi_{q,MB(i)} \\
=& -\Psi_{q,MB(i)} + \mathbb{E}_{y_i|\mathrm{MB}}\left[\Psi_{q,MB(i)}\right].
\end{aligned}$$

Therefore, the partial derivative of pseudo-log-likelihood with respect to rule weight W_q is computed by:

$$\frac{\partial \log P^*(\boldsymbol{y} \mid \boldsymbol{x})}{\partial W_q} = \sum_{i=1}^{n} \left\{ \mathbb{E}_{y_i|\mathrm{MB}}\left[\sum_j \mathbf{1}_{\{y_i \to G_q^{(j)}\}} d(G_q^{(j)})\right] - \sum_j \mathbf{1}_{\{y_i \to G_q^{(j)}\}} d(G_q^{(j)}) \right\}.$$

A.2.2 Calculation of number of ground formulas for Kinship datasets

We present a detailed calculation of the number of ground formulas considered by PSL in Kinship datasets as follows.

Given

- a first-order logical rule F_q containing $|\mathcal{I}_q^-|$ premise atoms, and
- a knowledge base containing $|\mathcal{E}|$ number of entities,

the number of variables in F_q is $|\mathcal{I}_q^-| + 1$.

PSL grounds each rule by substituting the variables with all possible entities. The number of ground formulas created by this logic rule F_q on the knowledge base is:

$$|\mathcal{E}|^{|\mathcal{I}_q^-|+1}.$$

Thus the overall ground formulas created by the rule set $\{F_q\}_{q=1}^m$ is:

$$\sum_{q=1}^{m} |\mathcal{E}|^{|\mathcal{I}_q^-|+1}.$$

Given the statistics of Kinship datasets in Table 3.2, rules statistics are shared across different sizes of Kinship datasets, each dataset contains 12 rules that contain two variables and 9 rules that contain 3 variables. The number of ground formulas considered by PSL is thus computed by:

$$12 \times |\mathcal{E}|^2 + 9 \times |\mathcal{E}|^3. \tag{A.4}$$

By applying the Eq. equation (A.4), we can get the ground formula number for each size of the Kinship dataset, as presented in Table A.2:

Table A.2: Number of ground formulas of Kinship datasets created by classical grounding method.

Kinship Size	S1	S2	S3	S4	S5
Number of ground formulas	1,373,601	10,853,976	35,798,376	74,671,320	172,162,935

A.3 Notations of NeuSymEA

Notation	**Description**
$\mathcal{G}, \mathcal{G}'$	The source and target knowledge graphs, respectively
$\mathcal{E}, \mathcal{E}'$	The sets of entities in $\mathcal{G}$ and $\mathcal{G}'$, respectively
$\mathcal{R}, \mathcal{R}'$	The sets of relations in $\mathcal{G}$ and $\mathcal{G}'$, respectively
$\mathcal{T}, \mathcal{T}'$	The sets of relational triplets in $\mathcal{G}$ and $\mathcal{G}'$, respectively
$\mathcal{O}$	The set of observed aligned entity pairs between two knowledge graphs $\mathcal{G}$ and $\mathcal{G}'$
$\mathcal{H}$	Set of unobserved entity pairs, i.e., $\mathcal{E} \times \mathcal{E}' \backslash \mathcal{O}$
$\boldsymbol{v}_{(e,e')}$	Binary indicator variable for an entity pair (e, e'), where $\boldsymbol{v}_{(e,e')} = 1$ indicates alignment
$w_{p,p'}$	Confidence score of a rule-inferred alignment based on paths p and p'
$p_w(\boldsymbol{v}_{(e,e')} \mid \mathcal{G}, \mathcal{G}')$	Probability distribution of the alignment indicator $\boldsymbol{v}_{(e,e')}$ given knowledge graphs $\mathcal{G}$ and $\mathcal{G}'$
θ	Parameters of the neural model
δ	Threshold to select positive pair from the symbolic model
$\eta(r)$	Relation pattern measuring the uniqueness of an entity through relation r

Table A.3: Notations

A.4 Algorithms of NeuSymEA

A.4.1 Pseudo-code of Explainer

Below is the pseudo-code of how the explainer generates supporting rules as interpretations for the query pair. It consists of two stages: searching reachable anchor pairs, and parsing rule paths as well as calculating rule confidences.

Algorithm 2 Generating Interpretations for the Queried Entity Pair with Weighted Rules

Inputs: Subrelation probabilities $p_{sub}(r \subseteq r'), p_{sub}(r' \subseteq r)$ for $r, r' \in \mathcal{R}$; Knowledge Graph pair $(\mathcal{G}, \mathcal{G}')$; Maximum rule length $\mathcal{L}$; Anchor pairs $\mathcal{A}$ with source-to-target mapping `S2T` and target-to-source mapping `T2S`; Query entity pair (e_q, e'_q)

Outputs: Ranked rules based on confidence

1. Search Reachable Anchor Pairs within Max Depth $\mathcal{L}$

$RN \leftarrow \text{BFS}(e_q, \mathcal{G}, \mathcal{L})$ /* Search reachable neighbors of e_q using breadth-first search, max depth $\mathcal{L}$ */

$RN' \leftarrow \text{BFS}(e'_q, \mathcal{G}', \mathcal{L})$ /* Search reachable neighbors of e'_q using breadth-first search, max depth $\mathcal{L}$ */

$RN_a \leftarrow RN \cup \texttt{T2S}(RN'; \mathcal{A})$ /* Find source nodes of reachable anchor pairs using hash mapping */

$RA \leftarrow \{(e, \texttt{S2T}(e; \mathcal{A})) \mid e \in RN_a\}$ /* Identify reachable anchor pairs */

2. Parse and Rank Rules Based on Confidence

for $\forall (e, e') \in RA$ **do**

Extract paths: $p(e, e_q) = r_1 \wedge r_2 \wedge \ldots$, $p'(e', e'_q) = r'_1 \wedge r'_2 \wedge \ldots$

if $|p(e, e_q)| \neq |p'(e', e'_q)|$ **then**

$w_{p(e,e_q),p'(e',e'_q)} \leftarrow 0$ /* If path lengths don't match, rule confidence is 0 */

else

$w_{p(e,e_q),p'(e',e'_q)} \leftarrow \prod_{i=1}^{|p|} \eta(r_i) \cdot \eta(r'_i) \cdot \frac{p_{sub}(r_i \subseteq r'_i) + p_{sub}(r'_i \subseteq r_i)}{2}$ /* Compute rule confidence by products of subrelation probabilities and relation functionalities */

end if

end for

Sort the rules (p, p') by $w_{p,p'}$ in descending order

Return the ranked rules =0

References

[1] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss Markov random fields and probabilistic soft logic. *Journal of Machine Learning Research*, 32(1), 2017.

[2] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*, 2019.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[4] Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*, 2023.

[5] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, page 1511–1517, 2017.

[6] Shengyuan Chen, Qinggang Zhang, Junnan Dong, Wen Hua, Qing Li, and Xiao Huang. Entity alignment with noisy annotations from large language models. *arXiv preprint arXiv:2405.16806*, 2024.

[7] Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. Label-free node classification on graphs with large language models (LLMs). In *The Twelfth International Conference on Learning Representations*, 2024.

[8] Keiwei Cheng, Nesreen K Amed, and Yizhou Sun. Neural compositional rule learning for knowledge graph reasoning. In *International Conference on Learning Representations*, 2023.

[9] Kewei Cheng, Nesreen K Ahmed, Ryan A Rossi, Theodore Willke, and Yizhou Sun. Neural-symbolic methods for knowledge graph reasoning: A survey. *ACM Transactions on Knowledge Discovery from Data*, 2024.

[10] Kewei Cheng, Jiahao Liu, Wei Wang, and Yizhou Sun. Rlogic: Recursive logical rule learning from knowledge graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 179–189, 2022.

[11] Lauren Nicole DeLong, Ramon Fernández Mir, and Jacques D Fleuriot. Neurosymbolic ai for reasoning over knowledge graphs: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[12] Junnan Dong, Qinggang Zhang, Xiao Huang, Keyu Duan, Qiaoyu Tan, and Zhimeng Jiang. Hierarchy-aware multi-hop question answering over knowledge graphs. In *Proceedings of the ACM Web Conference 2023*, pages 2519–2527, 2023.

[13] Junnan Dong, Qinggang Zhang, Xiao Huang, Qiaoyu Tan, Daochen Zha, and Zhao Zihao. Active ensemble learning for knowledge graph error detection. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 877–885, 2023.

[14] Junnan Dong, Qinggang Zhang, Huachi Zhou, Daochen Zha, Pai Zheng, and Xiao Huang. Modality-aware integration with large language models for knowledge-based visual question answering, 2024.

[15] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610, 2014.

[16] Huang Fang, Yang Liu, Yunfeng Cai, and Mingming Sun. MLN4KB: an efficient markov logic network engine for large-scale knowledge bases and structured logic rules. In *The International World Wide Web Conference 2023*, 2023.

[17] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 192–202, 2016.

[18] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[19] Fuzhen He, Zhixu Li, Yang Qiang, An Liu, Guanfeng Liu, Pengpeng Zhao, Lei Zhao, Min Zhang, and Zhigang Chen. Unsupervised entity alignment using attribute triples and relation triples. In *Database Systems for Advanced Applications: 24th International Conference, DASFAA 2019, Chiang Mai, Thailand, April 22–25, 2019, Proceedings, Part I 24*, pages 367–382, 2019.

[20] Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. Next-generation database interfaces: A survey of llm-based text-to-sql. *arXiv preprint arXiv:2406.08426*, 2024.

[21] Feiran Huang, Zefan Wang, Xiao Huang, Yufeng Qian, Zhetao Li, and Hao Chen. Aligning distillation for cold-start item recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 1147–1157, 2023.

[22] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 105–113, 2019.

[23] Xuhui Jiang, Yinghan Shen, Zhichao Shi, Chengjin Xu, Wei Li, Zixuan Li, Jian Guo, Huawei Shen, and Yuanzhuo Wang. Unlocking the power of large language models for entity alignment. *arXiv preprint arXiv:2402.15048*, 2024.

[24] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *The Semantic Web–ISWC 2011: 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I 10*, pages 273–288, 2011.

[25] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. Fast and exact rule mining with amie 3. In *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17*, pages 36–52. Springer, 2020.

[26] Freddy Lecue. On the role of knowledge graphs in explainable ai. *Semantic Web*, 11(1):41–51, 2020.

[27] Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*, 2023.

[28] Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. Zerog: Investigating cross-dataset zero-shot transferability in graphs. *arXiv preprint arXiv:2402.11235*, 2024.

[29] Zhixun Li, Xin Sun, Yifan Luo, Yanqiao Zhu, Dingshuo Chen, Yingtao Luo, Xiangxin Zhou, Qiang Liu, Shu Wu, Liang Wang, et al. GSLB: the graph structure learning benchmark. *Advances in Neural Information Processing Systems*, 36, 2024.

[30] Bing Liu, Harrisen Scells, Wen Hua, Guido Zuccon, Genghong Zhao, and Xia Zhang. Guiding neural entity alignment with compatibility. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 491–504. Association for Computational Linguistics, 2022.

[31] Xin Mao, Wenting Wang, Yuanbin Wu, and Man Lan. Boosting the speed of entity alignment 10×: Dual attention matching network with normalized hard sample mining. In *Proceedings of the Web Conference 2021*, pages 821–832, 2021.

[32] Xin Mao, Wenting Wang, Yuanbin Wu, and Man Lan. LightEA: A scalable, robust, and interpretable entity alignment framework via three-view label propagation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.

[33] Xin Mao, Wenting Wang, Huimin Xu, Yuanbin Wu, and Man Lan. Relational reflection entity alignment. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1095–1104, 2020.

[34] Zhiyuan Qi, Ziheng Zhang, Jiaoyan Chen, Xi Chen, Yuejia Xiang, Ningyu Zhang, and Yefeng Zheng. Unsupervised knowledge graph alignment by probabilistic reasoning and semantic embedding. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021.

[35] Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. *arXiv preprint arXiv:2010.04029*, 2020.

[36] Meng Qu and Jian Tang. Probabilistic logic neural networks for reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7710–7720, Vancouver, Canada, 2019.

[37] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Denny Zhou, Jure Leskovec, and Dale Schuurmans. Smore: Knowledge graph completion and multi-hop reasoning in massive knowledge graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1472–1482, 2022.

[38] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62:107–136, 2006.

[39] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in Neural Information Processing Systems*, 32, 2019.

[40] Tara Safavi and Danai Koutra. Codex: A comprehensive knowledge graph completion benchmark. *arXiv preprint arXiv:2009.07810*, 2020.

[41] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI conference on artificial intelligence*, pages 3060–3067, 2019.

[42] Chen Shengyuan, Yunfeng Cai, Huang Fang, Xiao Huang, and Mingming Sun. Differentiable neuro-symbolic reasoning on large-scale knowledge graphs. *Advances in Neural Information Processing Systems*, 36, 2024.

[43] Baoxu Shi and Tim Weninger. Proje: Embedding projection for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[44] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. Paris: Probabilistic alignment of relations, instances, and schema. In *Proceedings of the 38th International Conference on Very Large Databases*, pages 157–168, 2012.

[45] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *16th International Conference on the World Wide Web*, pages 697–706, 2007.

[46] Zequn Sun, Wei Hu, and Chengkai Li. Cross-lingual entity alignment via joint attribute-preserving embedding. In *Proceedings of the 16th International Semantic Web Confernece*, pages 628–644. Springer, 2017.

[47] Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, number 2018, 2018.

[48] Zequn Sun, Qingheng Zhang, Wei Hu, Chengming Wang, Muhao Chen, Farahnaz Akrami, and Chengkai Li. A benchmarking study of embedding-based entity alignment for knowledge graphs. *Proceedings of the VLDB Endowment*, 13(12):2326–2340, 2020.

[49] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[50] Xiaobin Tang, Jing Zhang, Bo Chen, Yang Yang, Hong Chen, and Cuiping Li. Bert-int:a bert-based interaction model for knowledge graph alignment. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3174–3180, 2020.

[51] Ilaria Tiddi and Stefan Schlobach. Knowledge graphs as tools for explainable machine learning: A survey. *Artificial Intelligence*, 302:103627, 2022.

[52] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*, 2019.

[53] Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. Knowledge graph prompting for multi-document question answering. In

Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 19206–19214, 2024.

[54] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, 2014.

[55] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 349–357, 2018.

[56] Xuansheng Wu, Huachi Zhou, Wenlin Yao, Xiao Huang, and Ninghao Liu. Towards personalized cold-start recommendation with prompts. *arXiv preprint arXiv:2306.17256*, 2023.

[57] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. Relation-aware entity alignment for heterogeneous knowledge graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

[58] Yue Xu, Hao Chen, Zefan Wang, Jianwen Yin, Qijie Shen, Dimin Wang, Feiran Huang, Lixiang Lai, Tao Zhuang, Junfeng Ge, and Xia Hu. Multi-factor sequential re-ranking with perception-aware diversification. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 5327–5337, 2023.

[59] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. *Advances in neural information processing systems*, 30, 2017.

[60] Qinggang Zhang, Junnan Dong, Hao Chen, Wentao Li, Feiran Huang, and Xiao

Huang. Structure guided large language model for sql generation. *arXiv preprint arXiv:2402.13284*, 2024.

[61] Qinggang Zhang, Junnan Dong, Hao Chen, Daochen Zha, Zailiang Yu, and Xiao Huang. Knowgpt: Knowledge injection for large language models, 2024.

[62] Qinggang Zhang, Junnan Dong, Keyu Duan, Xiao Huang, Yezi Liu, and Linchuan Xu. Contrastive knowledge graph error detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2590–2599, 2022.

[63] Qinggang Zhang, Junnan Dong, Qiaoyu Tan, and Xiao Huang. Integrating entity attributes for error-aware knowledge graph embedding. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–16, 2023.

[64] Qinggang Zhang, Keyu Duan, Junnan Dong, Pai Zheng, and Xiao Huang. Logical reasoning with relation network for inductive knowledge graph completion. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4268–4277, 2024.

[65] Wen Zhang, Jiaoyan Chen, Juan Li, Zezhong Xu, Jeff Z Pan, and Huajun Chen. Knowledge graph reasoning with logics and embeddings: survey and perspective. *arXiv preprint arXiv:2202.07412*, 2022.

[66] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. Efficient probabilistic logic reasoning with graph neural networks. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.

[67] Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34:19172–19183, 2021.

[68] Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*, 2023.

[69] Yu Zhao, Yike Wu, Xiangrui Cai, Ying Zhang, Haiwei Zhang, and Xiaojie Yuan. From alignment to entailment: A unified textual entailment framework for entity alignment. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8795–8806, 2023.

[70] Da Zheng, Xiang Song, Chao Ma, Zeyuan Tan, Zihao Ye, Jin Dong, Hao Xiong, Zheng Zhang, and George Karypis. Dgl-ke: Training knowledge graph embeddings at scale. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 739–748, 2020.

[71] Ziyue Zhong, Meihui Zhang, Ju Fan, and Chenxiao Dou. Semantics driven embedding learning for effective entity alignment. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 2127–2140, 2022.

[72] Huachi Zhou, Jiaqi Fan, Xiao Huang, Ka Ho Li, Zhenyu Tang, and Dahai Yu. Multi-interest refinement by collaborative attributes modeling for click-through rate prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, page 4732–4736, 2022.