Bulletin of the Technical Committee on

Data Engineering

March 2021 Vol. 44 No. 1

IEEE Computer Society

Letters

Special Issue on Learning and Reasoning on Knowledge Graphs and Applications

Conference and Journal Notices

The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TCDE web page is http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at
http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

The March issue of the Data Engineering Bulletin focuses on the intricate interplay as well as a significant gap between data management and machine learning when it comes to supporting real-life business applications.

The opinion piece of this issue features a group of distinguished researchers and their assessment and prognosis of machine learning's current and future roles in building database systems. Besides highlighting several specific potentials and challenges such as using machine learning to optimize database indices and query optimization, the article also gives a great overview of how databases, data analytics and machine learning, system and infrastructure, work together to support today's business needs. It is clear that the business needs, the volume, velocity, and variety of the data, the latency and throughput requirements have evolved dramatically and in consequence, data management systems must adapt. The opinion pieces described four disruptive forces underneath the evolution, which are likely to influence future data systems.

Our associate editor Sebastian Schelter put together the current issue–Data Validation for Machine Learning Models and Applications–that consists of six papers from leading researchers in industry and academia. The papers focus on data validation, which is a critical component in end-to-end machine learning pipelines that many business applications rely on.

Haixun Wang
Instacart

Letter from the Special Issue Editor

Software applications that learn from data using machine learning (ML) are being deployed in increasing numbers in the real world. Designing and operating such applications introduces novel challenges, which are very different from the challenges encountered in traditional data processing scenarios. ML applications in the real world exhibit a much higher complexity than "text book" ML scenarios (e.g., training a classifier on a pre-existing dataset). They do not only have to learn a model, but must define and execute a whole ML pipeline, which includes data preprocessing operations such as data cleaning, standardisation and feature extraction in addition to learning the model, as well as methods for hyperparameter selection and model evaluation. Such ML pipelines are typically deployed in systems for end-to-end machine learning, which require the integration and validation of raw input data from various input sources, as well as infrastructure for deploying and serving the trained models. The system must also manage the lifecycle of data and models in such scenarios, as new (and potentially changing) input data has to be continuously processed, and the corresponding ML models have to be retrained and managed accordingly. The majority of these challenges have only recently begun to attract the attention of the data management community. A major obstacle is that the behavior of ML-based systems heavily depends on the consumed input data, which can rapidly change, for example due to changed user behavior or due to errors in external sources that produce the inputs. This area represents a gap between the data management and ML communities: research in ML mostly focuses on learning algorithms, and research in data management is mostly concerned with data processing and integration. In this issue, we focus on this gap in data validation for machine learning, and provide perspectives from both the academic and industrial research communities to learn about the state of the art, open problems and to uncover interesting research directions for the future.

The first paper presents *A Data Quality-Driven View of MLOps* and demonstrates how different aspects of data quality propagate through various stages of machine learning development. It connects data quality to the downstream machine learning process, an approach that is also taken by our second paper, which argues that we should move *From Cleaning before ML to Cleaning for ML*. The authors propose an end-to-end approach to take the entire application's semantics and user goals into account when cleaning data, instead of performing the cleaning operations in an isolated manner beforehand.

The next two papers on *Validating Data and Models in Continuous ML pipelines* and *Automated Data Validation in Machine Learning Systems* from Google and Amazon provide us with an industry perspective on the area in the focus of this issue. The first paper describes tools developed at Google for the analysis and validation of two of the most important types of artifacts: Datasets and Models. These tools (which are part of the Tensorflow Extended Platform) are currently deployed in production at Google and other large organizations, and are heavily inspired by well-known principles of data-management systems. The second paper from Amazon reviews some of the solutions developed to validate data at the various stages of a data pipeline in modern ML applications, discusses to what extent these solutions are being used in practice, and outlines research directions for the automation of data validation.

The subsequent paper on *Enhancing the Interactivity of Dataframe Queries by Leveraging Think Time* focuses on the highly exploratory and iterative nature of data validation in the early stages of an ML application, where data scientists start with a limited understanding of the data content and quality, and perform data validation through incremental trial-and-error. The final paper of this issue on *Responsible AI Challenges in End-to-end Machine Learning* completes the view on data validation for machine learning by connecting it with pressing issues from the area of responsible data management.

Working on this issue has been a privilege for me, and I would like to thank the authors for their contributions.

Yangqiu Song
The Hong Kong University of Science and Technology

utf8]inputenc

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Logical Queries on Knowledge Graphs: Emerging Interface of Incomplete Relational Data

Zihao Wang[†] Hang Yin[‡] Yangqiu Song[†]
[†] Department of CSE, HKUST, Hong Kong, China {zwanggc,yqsong}@cse.ust.hk
[‡] Tsinghua University, Beijing, China h-yin20@mails.tsinghua.edu.cn

Abstract

As a graph abstraction of real-world relational data, knowledge graphs contain large amounts of factual knowledge but suffer from incompleteness. Such incompleteness, also known as the open-world assumption, prevents logical queries from being answered through the query-answering paradigms for relational databases. In this paper, we discuss how learning-based models bridge incomplete relational data and logical queries from the perspective of rigorous formal logic. Existing works are structured as a three-party game among knowledge graphs, logical queries, and learning-based models. Compared to the well-established tale between databases and queries in the database theory, current achievements are still preliminary. Our work pointed out several possible directions for future work.

1 Introduction

The knowledge graph is perhaps one of the simplest representations of relational knowledge [8, 54, 38]. A knowledge graph contains a large number of triples, where each triple (h, r, t) contains a head entity h, a tail entity t, and the relation r in between. Such a simple format enables factual knowledge to be adapted to answer real-world questions [47, 59, 45, 40, 32].

Under the view of data engineering, a knowledge graph is no more than a (relational) database [1]. It is natural to expect first-order queries, a fundamental type of relational database queries that have been well-understood theoretically [30] and efficiently solved practically [29], can also be answered on knowledge graphs. However, applying the existing query-answering algorithms designed for (relational) databases can not address logical queries on knowledge graphs. The major obstacle is that large-scale knowledge graphs are inherently incomplete. Modern large-scale knowledge graphs are constructed by crowd-sourcing [54] or even by automatic information extraction pipelines [11]. Given an observed knowledge graph, the non-existence of a triple does not imply that such a triple is not part of the underlying knowledge graph. This issue is acknowledged as the open-world assumption [31]. Therefore, the actual answers to logical queries are not guaranteed by running existing query-answering algorithms on knowledge graphs.

The issue of missing knowledge founds its partial solution with learning-based models. Models that learn from the existing observed triples can predict the missing triples in the underlying knowledge graph. The research topic focusing on the prediction of missing triples with learned embeddings is known as knowledge graph representation or knowledge base completion [25, 44]. Then, the logical queries can be addressed by combining the query-answering algorithms and the learning-based models [42, 28, 16]. The participation of learning-based models brings unique strengths and weaknesses to logical query answering, especially compared to traditional query answering algorithms for relational databases [42]. For traditional databases, query-answering algorithms are discussed from the data perspective and the query perspective. The complexity of such algorithms

is characterized by the data complexity (the complexity grows with the size of the database) and the query complexity (the complexity grows with the size of the query), respectively. When answering logical queries with learning-based models, it is essential to understand how learning-based models interact with knowledge graphs (model-data interaction) and logical queries (model-query interaction), besides how to design, train, and infer such models.

In this paper, we review recent advancements in logical query answering on knowledge graphs. Our introduction begins with how to define the problems and construct datasets and then to learning-based query answering methods. Our discussion is organized from three perspectives: knowledge graphs (data) perspective, logical query perspective, and model perspective. Section 2 introduces the logical query answering tasks on knowledge graphs in the rigorous language of model theory. All existing datasets and benchmarks are discussed in Section 3 based on the rigorous definitions. Section 4 discusses the learning-based methods for knowledge graph logical query answering. Existing approaches are categorized by their interaction with the knowledge graph (data perspective) and the logical queries (query perspective). Section 5 discusses the limitation of the existing work and several possible future directions.

2 Preliminaries for Logical Queries

Logical queries are formally defined by $\mathcal{L}$-formula on an $\mathcal{L}$-structure, which are central concepts in model theory [36]. Specifically, $\mathcal{L}$-structure rigorously justifies the concept of knowledge graphs while $\mathcal{L}$-formula formally defines the scope of queries. Formal definitions provide a framework to investigate the data and the query aspects of the knowledge graph logical query answering tasks.

The minimal and informal description of knowledge graphs is a set of triples $\{(h_i, r_i, t_i)\}$ that encapsulates the real-world knowledge, where $h_i, t_i \in \mathcal{E}$ are the head and tail entities and $r_i \in \mathcal{R}$ is the connected relation. Formally speaking, a knowledge graph is an $\mathcal{L}_{KG}$-structure defined by a language $\mathcal{L}_{KG}$ for knowledge graphs. The concept of language and structure, which concerns syntax and semantics of first-order logic respectively, is defined in Definition 1 and Definition 2.

Definition 1 (Language): A language $\mathcal{L}$ is a triple $(\mathcal{E}, \mathcal{R}, \mathcal{F})$ where $\mathcal{E}$ is the set of constant symbols, $\mathcal{R}$ is the set of predicate symbols, and $\mathcal{F}$ is the set of function symbols. For each $r \in \mathcal{R}$ and $f \in \mathcal{F}$, associated integers n_r and n_f indicate the number of inputs for R and F, respectively.

All (classic) knowledge graphs can be described by a language $\mathcal{L}_{KG}$ by letting (1) $\mathcal{R}$ be the set of binary relations ($n_r = 2$), (2) the constant set $\mathcal{E}$ is finite and contains all entities, and (3) $\mathcal{F} = \emptyset$. $\mathcal{E}$ and $\mathcal{R}$ are finite sets without further classification.

Definition 2 ($\mathcal{L}$-structure): An $\mathcal{L}$-structure $\mathcal{M}$ for the language $\mathcal{L}$ is a pair (M, I),where M is a non-empty set called the universe of $\mathcal{M}$ and I is an interpretation function that: (i) For each $c \in \mathcal{E}, I(C) \in M$; (ii) For each $r \in \mathcal{R}, I(r) \subset M^{n_r}$, where n_r is the size of relations; (iii) For each $f \in \mathcal{F}$, a function $I(f) : M^{n_f} \mapsto M$, where n_f is the number of inputs in f.

A knowledge graph $\mathcal{KG}$ induces an $\mathcal{L}_{KG}$-structure by letting $M = \mathcal{E}$ (with $\forall e \in \mathcal{E}, I(e) = e$), so that each relation $I(r)$ is a subset of M^2. The open world assumption of knowledge graphs can then be formally stated in Definition 3.

Definition 3 (Open world assumption): For a given language $\mathcal{L}_{KG}$ of knowledge graph, there is an **u**nderlying $\mathcal{L}_{KG}$-structure $\mathcal{KG}_u$, and one (or more) **o**bserved $\mathcal{L}_{KG}$-structure (s) $\mathcal{KG}_o$, such that for each relation $r \in \mathcal{R}$, the relation set of $\mathcal{KG}_o$ and $\mathcal{KG}_u$ satisfies $I_{\mathcal{KG}_o}(r) \subseteq I_{\mathcal{KG}_u}(r) \subseteq M^2$.

Then, it is ready to define the logical queries with formal concepts, including $\mathcal{L}$-terms and $\mathcal{L}$-formulae. The current discussion focuses on first-order logic.

Table 1: Summation of open source knowledge graphs used in the existing datasets. KGs are sorted by the number of entities.

Knowledge graph	# Entities	# Relations	# Total edges	Comment	Datasets
FB15k-237 [9]	14,505	237	310,079	-	[42, 43, 19, 23, 55]
FB15k [51]	14,951	1,345	592,213	-	[42, 43, 55]
DBPedia [5]	34,575	3	240,942	Hierarchical KG	[14]
WN18RR [37]	40,903	11	103,509	Hierarchical KG	[24]
NELL955 [56]	63,361	200	142,804	-	[42, 43, 23, 55]
DRKG [61]	97,238	107	5,874,271	-	[13]
FB400k [48, 8]	409,829	918	2,151,671	-	[41]
ogbl-wikikg2 [22]	2,500,604	535	17,137,181	-	[41, 19]
Freebase [8]	86,054,151	14,824	338,586,276	-	[41]

Definition 4 ($\mathcal{L}$-term): $\mathcal{L}$-terms be the smallest set of $\mathcal{T}$ such that: (i) $e \in \mathcal{T}$ for each constant symbol $e \in \mathcal{E}$; (ii) Any variable $v_i \in \mathcal{T}$ for 1, 2, ... ; (iii) if $t_1, t_2, ..., t_{n_f} \in \mathcal{T}$ and $f \in \mathcal{F}$, then $f(t_1, ..., t_{n_f}) \in \mathcal{T}$.

Definition 5 (Atomic $\mathcal{L}$-formula): ϕ is an atomic $\mathcal{L}$-formula if ϕ is $r(t_1, ..., t_{n_r})$, where $r \in \mathcal{R}$ is a relation symbol and $t.$ are $\mathcal{L}$-terms. Given $\mathcal{L}$-structure $\mathcal{M}$, we say $\mathcal{M} \models r(r_1, ..., r_{n_r})$ is True if and only if the tuple $(t_1, ..., t_{n_r}) \in I(r)$.

Definition 6 ($\mathcal{L}$-formula): The set of $\mathcal{L}$-formula is the smallest set Φ containing all atomic formulae such that: (i) if $\phi \in \Phi$, then $\neg\phi \in \Phi$; (ii) if $\phi, \psi \in \Phi$, then $(\phi \wedge \psi) \in \Phi$ and $(\phi \vee \psi) \in \Phi$; (iii) if $\phi \in \Phi$ and v_i is any variable, then $\exists v_i \phi \in \Phi$ and $\forall v_i \phi \in \Phi$.

We say a variable v is *free* if there are no associated quantifiers. Otherwise, it is *bounded*. We use $\phi(v)$ indicates the $\mathcal{L}$-formula ϕ contains a free variable v. An $\mathcal{L}$-formula without free variables is called an $\mathcal{L}$-sentence. Logical queries are $\mathcal{L}$-formulae and can be evaluated given a knowledge graph $\mathcal{KG}$. As studied in the database literature [53], we discuss boolean queries, set-valued queries, and aggregate queries.
A **boolean query** is an $\mathcal{L}_{KG}$-sentence s. Given the $\mathcal{L}_{KG}$-structure $\mathcal{M}$, its answer is the boolean value indicating whether $\mathcal{M} \models s$ or not.
A **set-valued query** is an $\mathcal{L}_{KG}$-formula ϕ with exactly one free variable v. Given the $\mathcal{L}$-model $\mathcal{M}$, its answer set is $\{e : e \in \mathcal{E}, \mathcal{M} \models \phi(v = e)\}$.
An **aggregate query** looks for the aggregation statistics of the answer set of a set-valued query, such as counting, summation, and so on.

3 Logical Query Answering Datasets and Evaluations

Based on the formal concepts given in Section 2, we are ready to formally justify existing datasets from the knowledge graphs (the data perspective) that are queried and the scope of logical queries (the query perspective) to be answered. Instead of going through all existing datasets, we discuss the underlying knowledge graphs and logical queries.

3.1 Knowledge Graphs

Table 1 shows the knowledge graphs investigated in existing datasets. Existing evaluation includes knowledge graphs of different scales (from 10^3 to 10^9) and various properties. Notably, the knowledge graphs with

Table 2: Formal definitions of three typical query families. Compared to the first-order logic formula defined formally with Definition 6, three query families are defined using a subset of connectives or quantifiers (indicated by ✓).

Query Family	$\wedge$	$\vee$	$\neg$	$\exists$	$\forall$
Conjunctive Query (CQ)	✓	✗	✗	✓	✗
Existential Positive First Order (EPFO)	✓	✓	✗	✓	✗
Existential First Order (EFO)	✓	✓	✓	✓	✗

hierarchical relation [37, 5] contain the prior and more complicated structures on entities and relations, respectively, which could be leveraged to improve the performance of query answering method. Inductive logical query answering [19] also considers an inductive KG setting [49], which is more challenging under the open-world assumption.

3.2 Logical Queries

The scope of queries that can be answered is not rigorously justified in the existing works. Though some works claim they are addressing the first-order queries [43] or logical queries in general [33], their definitions are still a strict subset of the first-order queries defined in Definition 6. Table 2 summarizes the formal definitions for three typical query families. The scopes of three query families are sorted in increasing order, i.e., CQ $\subseteq$ EPFO $\subseteq$ EFO. The universal quantifier $\forall$ is not widely discussed because its semantics is not usually interpretable over the knowledge graphs. This is the same order for query families that learning-based query answering methods tried to solve.

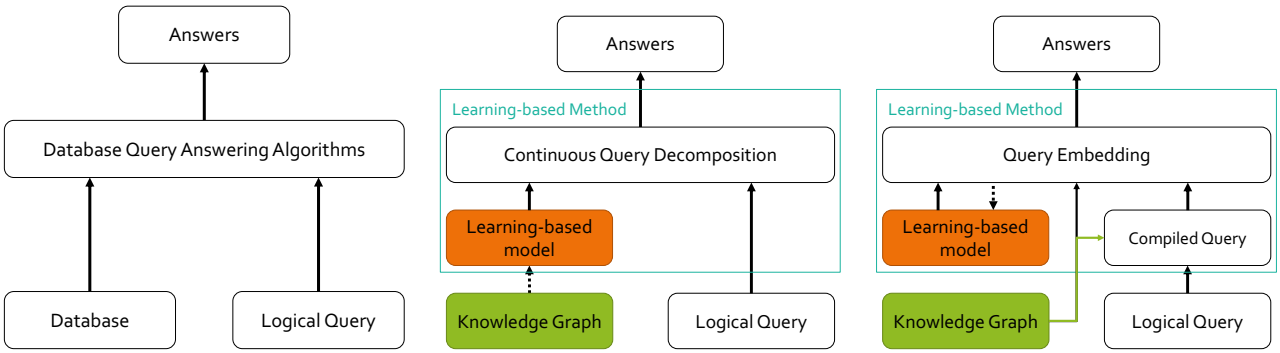
However, additional assumptions are made so that queries are more suitable to be answered by learning-based methods. Common assumptions include that (1) the query is assumed to be acyclic and (2) only one free variable in the logical formula. The first assumption skips the queries that may involve hard constraint satisfaction problems and the second assumption restricts the space complexity of search within $O(|\mathcal{E}|)$. These assumptions are not necessary for all learning-based models. For example, BiQE [28] accepts conjunctive queries with multiple free variables, and CQD [4] answers general EPFO queries without any additional assumptions. Most existing datasets are usually constructed as companions to evaluate the proposed methods, especially query embedding methods with the operator tree representation discussed in Section 4.2.1. Such methods assume the answers with respect to the only free variable can be computed by executing learnable set operators. The execution order of set operators is decided by compiling the query formula into an operator tree. Thus, popular datasets [42, 43, 55] for EPFO and EFO queries also accept implicit assumptions: (1) the only free variable v is assumed to be the root of the set operator tree; (2) all leaf nodes must be constant entity symbols rather than existential variables.

We see that these assumptions are not related to the logic but only to make the queries easier to solve by a special class of learning-based methods. There are no serious empirical evaluations for queries without such assumptions.

3.3 Evaluation Metrics

The query families are categorized by the formulation of logical formulae. Moreover, for each family, it is also able to construct three types of queries. Here we list popular evaluating metrics.

- For **boolean query**, this becomes a classification problem, and thus standard classification metrics can be applied, like ROC AUC score [43, 21].



(a) Database Query Answering (b) Continuous Query Decomposition (c) Query Embedding

Figure 1: Paradigms for logical query answering on knowledge graphs. The solid line indicates the query answering procedures, and the dashed lines indicate the procedures to obtain the model.

Table 3: Comparison between CQD and QE methods with their training data, query answering algorithms, and solvable queries. † indicates that the actual solvable queries may not be the formally defined logical query family.

Methods	Training Data	Query Answering Algorithms	Solvable Queries
Continuous Query Decomposition (CQD)	KG triples	Model + Optimization	EPFO
Query Embedding (QE)	QA samples	Operator Tree + Model	EFO-1 †
	QA samples	Query Graph + Model	EPFO †
	QA samples	Sequence + Model	EPFO †

- For **set-valued query**, standard metrics include Mean Reciprocal Rank (MRR) [42], Adjusted Mean Rank Index (AMRI) [2], Average percentile rank (APR) [21], and Hits at K (H@K) [42].
- For **aggregate query**, counting is the only problem that has been considered. Spearman's rank correlation coefficient [43] and Pearson's correlation coefficient [35] have been proposed when the model is incapable of predicting the precise number. Otherwise, Mean Absolute Error (MAE) is used [35].

4 Learning-Based Methods

Traditional database query answering algorithms are executed over the raw data to find the answers, see Figure 5(b). This paradigm fails in the cases of knowledge graph queries, where the underlying data is not observed. Current knowledge graph logical query answering methods address the open-world assumption with learning-based methods, see the blue boxes in Figure 1(b) and 1(c). In this paper, we categorize the learning-based methods into two categories by how they process the queries. The first category is Continuous Query Decomposition (CQD) and the second is Query Embedding (QE). We see that learning-based models are essential for both the CQD method and the QE method. The participation of the learning-based models adds an intermediate layer between incomplete knowledge graphs and logical queries.

The key to understanding learning-based methods is not only how to define the model but also how models interact with the knowledge graph data and logical queries. The model-query and model-data interaction jointly define the scope of queries that can be answered by each method. Specifically, the model-data interaction refers to how to obtain the model from data, including both training data and algorithms, and the model-query interaction refers to how learning-based models are combined with query answering algorithms. The model-data and model-query interaction are indicated by the dashed arrows and the arrows in Figure 1(b) and 1(c), respectively.

The rest of this section presents the details of these two types of methods. Table 3 compares these two types from three aspects, including training data (model-data interaction), query answering algorithms (model-query interaction), and queries that can be solved. The most distinct difference between CQD and QE is how they treat the model as part of the query answering algorithms. Continuous query decomposition estimates the answer embedding by solving an optimization problem with continuous objective induced from the original logical formula with logical t-norms [20]. This objective, computed by the result of the model inference, will be evaluated multiple times during the optimization process. Therefore, the model will be inferred multiple times. This formulation is denoted as Model + Optimization in Table 3 where the term Model is placed before Optimization. In contrast, query embedding methods first compile the logical queries into various representations such as operator trees, query graphs, or sequences. Then, the answer embedding, or the embedding of logical query, is estimated by simple forward inferring the model only once. This formulation is denoted as X + Model, where the term Model is placed after X indicating one of the operator trees, query graphs, or sequences. As natural consequences of such distinct differences in query answering algorithms, CQD and QE require different types of training data to obtain the model and can solve different types of queries. Moreover, the solvable query families for QE methods differ when the queries are compiled into different representations. We note that the solvable queries are confirmed in existing works, and they can be of course extended to broader classes.

The Disjunctive Normal Form (DNF) [36] of the logical queries is of particular interest. This is because (1) all first-order queries can be converted to DNF, and (2) The answer set of DNF existential queries can be regarded as the union of the answer sets of its containing conjunctive clauses. For example, EPFO queries can be answered by compositing the solutions of conjunctive queries. Similarly, general existential logical queries can be answered once the queries with conjunction and negation can be well answered.

4.1 Continuous Query Decomposition (CQD)

The model used in CQD [4] method is no more than a neural link predictor, which is the key part of knowledge graph representaiton [62, 25]. Specifically, CQD leverages the pretrained ComplEx embedding [52] and achieves state-of-the-art performances in EPFO queries on dataset [42]. The training of the neural link predictor is not related to the logical queries but only knowledge graphs themselves. Recent survey [25] summarizes the construction of neural link predictors and how to train neural link predictors well is discussed in [44].

CQD uses neural link predictors to solve EPFO queries with a search algorithm. It considers the EPFO query $\phi(v)$ in the following DNF form.

$$\phi(v) = \exists x_1, ..., x_n, (r_{11} \wedge \cdots \wedge r_{1k_1}) \vee \cdots \vee (r_{l1} \wedge \cdots \wedge r_{lk_l}), \tag{1}$$

where r_ij is the atomic formula (without negation) and v is the only free variables associated to one or more r_ij.

Once the neural link predictor is obtained, each atomic formula $r(h, t)$ can be evaluated with its probability $p(h, r, t)$ of being true. Let p_{ij} be the probability of ij-th triple, $\top$ and $\bot$ be the t-norm and t-conorm [20], we rewrite the EPFO query in Equation (1) into the optimization problem:

$$\max_{v,x_1,...,x_n} (p_{11} \top \cdots \top p_{1k_1}) \bot \cdots \bot (p_{l1} \top \cdots \top p_{lk_l}). \tag{2}$$

The objective in Problem (2) is continuous since p, $\top$, and $\bot$ is continuous. Embedding of variables $v, x_1, ..., x_n$ will be optimized over the continuous embedding space with gradient-based optimization or searched on the discrete set $\mathcal{E}$ with beam-search. Therefore, complex structures of logical queries are handled by solving the optimization problem with the compositional objective function. CQD is also applicable to EPFO queries with multiple free variables in principle. However, this is not empirically justified because of the lack of corresponding datasets. The drawback of CQD is also apparent: CQD eventually estimates the embedding of variables, which remains a significant gap in answering aggregate queries.

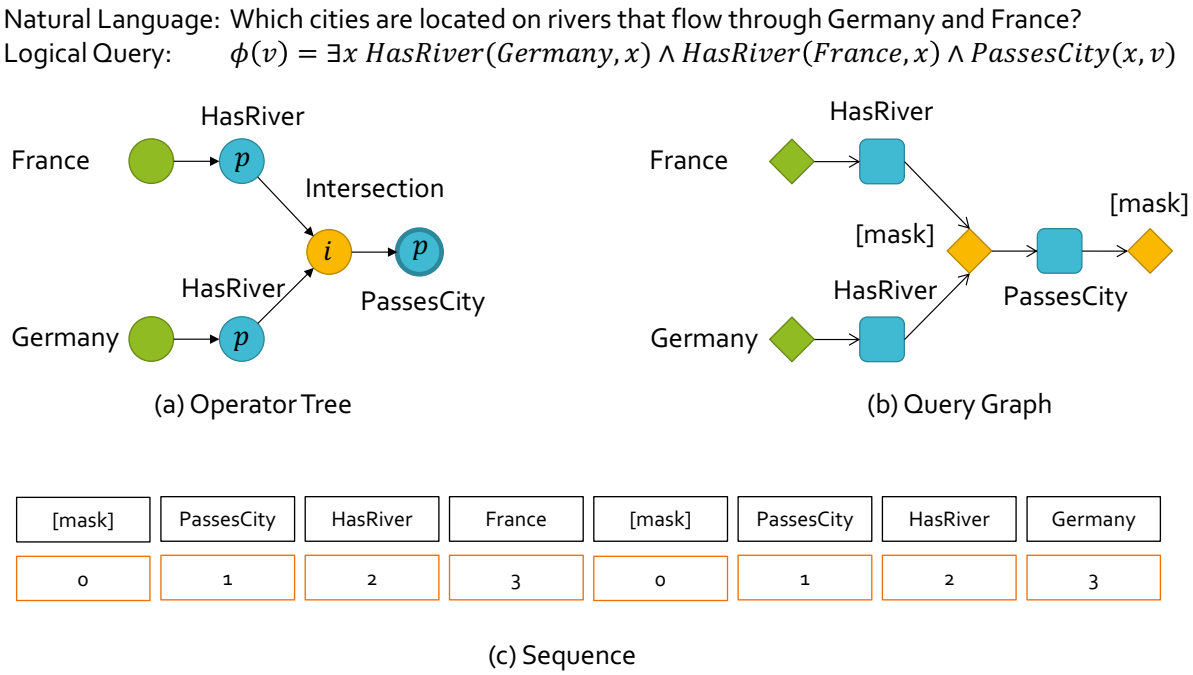


Figure 2: Three ways to represent logical query in the query embedding methods. The example is taken from [28]. We see the logical query is converted into three formats, including (a) operator tree [55], where each node is a set operator; (b) query graph [34], where the nodes are terms and predicates; (c) sequences [28], where a logical query is transformed as the concatenation of several multi-hop queries with a special positional encoding.

4.2 Query Embedding (QE)

In contrast to CQD, query embedding methods use explicit representations of the query structures. Direct inference of the learning-based model over specific query representation estimates the embedding of the target variable, i.e., the query embedding. Then the answers can be retrieved according to the query embedding. The general advantages and disadvantages of QE methods are also clear. On the one hand, models are inferred only once to obtain the answers. On the other hand, specific representation restricts the scope of solvable queries. In the current stage, there is no dominant representation of the queries. Figure 2 summarizes popular choices including operator trees [55], query graphs [34], and sequences [28]. Then we categorize different query embedding methods by their query representations and discuss each type from four perspectives: query representation, model design, training, and query answering algorithms.

4.2.1 Logical Queries as Operator Trees

Operator trees materialize the procedure of searching for answers as the execution of set operators, including set projection, intersection, union, and complement (with respect to a universe). Each entity itself is regarded as the simplest singleton, a set with only one element. The set projection is converted from relations by conducting a Skolemization [36] process of a logical formula. Other set operations such as intersection, union, and difference or negation are naturally induced by logical operators. Only a subset of first-order formulae can be converted into the operator trees because such a representation of logical queries imposes additional assumptions to Definition 6, as discussed in [55]. Meanwhile, the choice of set operators is not identical. For example, the set complement can be replaced by the set difference [33], and the set union can be replaced by intersection and vice versa with the set complement, according to De Morgan's laws. The learning-based model for operation tree representation

is designed to simulate the sets in the continuous embedding spaces and the set simulations with differentiable computations (usually with neural networks). Then we discuss the spaces used to represent sets and the modeling of the set operations.

Geometric Region Embeddings. Sets, as a collection of discrete objects for logical query answering, are naturally related to the geometrical region of some spaces. Compared to complex variable embeddings as vectors in CQD, embedding sets as geometric regions provides a straightforward estimation of the size of the answer set. The chosen forms of geometric regions are usually simple to be parameterized, such as the boxes used by Query2Box [42] and NewLook [33], sector cones two-dimensional spaces used by ConE [64], and hyperboloid used by HypE [14]. The set projection is usually modeled by neural networks that map the parameterized regions to new ones. Intersection and union operators in NewLook are modeled by some variants of deep set network [60] to merge multiple regions into one. Notably, the design of ConE [64] allows a closed-form set complement with sector cones, while NewLook [33] models the set difference between boxes with neural networks. How to handle negation operation using hyperboloid is not yet discussed.

Probabilistic Distribution Embeddings. Another thread of work models sets with the probabilistic distributions. Parameterized probabilistic distribution families include Beta distribution in BetaE [43], Gaussian distribution in PERM [13], and Gamma distribution in GammaE [58]. Probabilistic distributions are more sophisticated than geometrical regions in set representation because the techniques to construct new probabilistic distributions can be applied to model set operators. For example, a mixture of probabilistic distributions can be used to construct the union operation [13, 58]. Also, altering one parameter to its reciprocal is straightforward to represent the set complement for some family of distributions [43, 58]. Meanwhile, the normalized product of the Probability Distribution Functions (PDF) of Gamma distribution is also Gamma distribution. This fact enables GammaE to simplify the design of the intersection operator. Well-defined distances and divergences between distributions are powerful tools for defining the distances between the answer sets, such as Mahalanobis distance [13] and KL-divergence [43, 58]. Following the intuition of probabilistic distributions, LinE [24] considers the histogram as the unnormalized discretization of arbitrary PDF, and Query2Particles [6] handles a set of empirical samples (particles) from an arbitrary PDF. Such designs sacrifice parts of the closed-form properties but enlarge the family of underlying distributions with non-parametric representations.

Fuzzy Vector Embeddings. Fuzzy vector space [27] brings supreme advantages in modeling the set operations. Set intersection, union, and negation can be precisely modeled with fuzzy logical t-norms and t-conorms [20] by embedding sets as fuzzy vectors [35, 12], i.e., vectors in $[0, 1]^d$ where d is the dimension. The one thing left is to model the set projections with neural networks. FuzzQE [12] models the set projection by relational base decomposition [46] and LogicE [35] uses simple multi-layer perceptron. In addition, LogicE [35] models the lower and the upper bounds of the fuzzy logic truth value, which quantifies the uncertainty of each embedding to answer the aggregate query.

In contrast to various designs for embedding spaces, MLPMix [3] just embedded queries as the vectors in the Euclidean space. Such a simple embedding space does not guarantee any set properties, so it requires more advanced neural networks [50] to model set operators. Notably, MLPMix still has strong performance with simple embedding space compared to its predecessors. This might suggest that even the simplest Euclidean space is large enough. The overall performance results from the trade-off between set embeddings and set operations.

Neural symbolic methods also shed light on another way to improve performances. "Neural" indicates methods answering queries in low-dimensional embedding spaces and neural networks, while "symbolic" indicates referring to the original symbolic knowledge graphs. GNN-QE [65] estimates the probabilities of all entities given the set projection with graph neural network, NBFNet [66] over the observed knowledge graph. ENeSy [57] ensembles the neural model prediction and symbolic model prediction at each set projection, where the symbolic part is conducted by the TensorLog [15]. These methods show outstanding performances compared to the neural query embedding methods. However, scalability is always an issue since the intermediate states will unavoidably grow with the size of the underlying knowledge graph.

While various models for operator trees were proposed, the method to train the model is almost the same

as the first proposed to train Query2box [42]. Let $D(e, q)$ be the distance function between the entity e and the query embedding q, the objective function is

$$\ell_\theta(a, q) = -\log \sigma(\gamma - D(a, q)) - \frac{1}{k}\sum_{i=1}^{k} \log \sigma(D(v_i, q) - \gamma), \tag{3}$$

where $\sigma = \frac{1}{1+e^{-x}}$ is the sigmoid function, γ is the margin, a is an arbitrary answer, and v_i are k negative sampled answers.

4.2.2 Logical Queries as Query Graphs

Query graph presentation represents the terms and predicates as nodes and edges [16, 34]. Query graph representation does not assume the acyclicity as operator tree representation. Therefore it represents a larger set of logical queries. However, query graphs are not related to the disjunctive and negation connectives. The disjunction can be handled by converting existing queries to DNF queries (without negation), whose answer set is the combination of all conjunctive queries that can be addressed in the query graphs. How to represent the negation in the query graph has not been discussed yet.

The learning-based models for query graphs are Graph Neural Networks (GNN). MPQE [16] uses message passing networks and kgTransformer [34] uses graph transformers whose multi-layer perceptrons are upgraded by a Mixture of Experts (MoE) architecture. Besides the negative sampling objectives shown in Equation (3), kgTransformer proposed to first pretrain the model on sampled subgraphs and then finetune the model with the negative sampling loss on query answering data.

4.2.3 Logical Queries as Sequences

The last type of representation is to represent the queries in sequences. BiQE [28], see Figure 2 (c), proposes to convert the query graph to multiple paths from the answer node to the anchor entity nodes. Compared to the query graphs, the sequence representation even neglects the existential variables, only the anchor entity and the relations on its path to the answer node remain. Relative positional embeddings are also used to describe the distance from each token to the answer node. This representation also focuses on conjunctive queries, which do not contain disjunction and negation. Queries with disjunctions can be handled by combining conjunctive queries in the DNF. However, it is still not known how to handle the negation operation.

Once the sequence is defined, the sequence can be put into the transformer models. The output embedding of the `[mask]` token is aggregated to obtain the answer. Unlike the query graph representation that enforces the query structures explicitly, transformers could learn implicit logical structures via the self-attention mechanism.

5 Future Directions

Research about knowledge graphs logical query answering, though rapidly developed in recent years, is still limited compared to the well-established study about query answering on the relational database. For example, the query types that can be answered by the model are strict fragments of first-order queries. In this section, we discuss what is left in current research compared to logical query answering on incomplete relational data, which is the ultimate goal of this line of research. Formal definitions from model theory in Section 2 picture the gap between existing works and the ambitious goal. we summarize the limitations and discuss possible future directions from the data, query, and model perspectives.

5.1 Towards General Relational Data

A relational language $\mathcal{L}$ can describe general relational data. However, the language $\mathcal{L}_{KG}$ for knowledge graphs is defined with three additional assumptions, see Section 2. By rethinking three assumptions about $\mathcal{L}_{KG}$ and the open world assumption, we could extend the existing data model of knowledge graphs.

KG with Relations of $n_r > 2$. Knowledge graphs with n-ary relations [63] relax the first assumption that $n_r = 2, \forall R \in \mathcal{R}$ by accepting the relations whose $2 \leq n_r \leq n$. Thus, each element of the knowledge graph now is an $n+1$ triple list that contains n entities and one relation. The concept of n-ary relation is also related to the hyper-relation [18, 2] and the knowledge graph is extended from the directed graphs to hypergraphs.

KG with Attributes. Attributes expand the constant set $\mathcal{E}$ from all entities to *entities with attribute values*, thus relaxing the second assumption. Typical attributes include entity types [5, 23], triple timestamps [26, 45], and triple probabilities [11]. Additional attributes can be described in the n-ary relations. For example, the timestamp t associated with a triple (s, p, o) can be described with 4-triples (s, p, o, t). New attributes bring new predicates, enlarge the universe set, and enrich the query semantics. Two timestamps can be compared by "before", "equal", and "after", and can be composed to a time interval, which also introduces new predicates such as "during". The triple probabilities, as discussed in probabilistic databases [53], enable the estimation of the probabilities for boolean queries, the probability for each answer entity for set-valued queries, and the total probability for aggregate queries.

KG with Functions. Introducing attributes also makes it possible to define functions between attribute values. Taking the timestamp attribute as an example, new timestamps can be computed by a time movement function with a timestamp and a time interval as inputs. These features are rigorously defined in pure symbolic systems [17]. It is worth discussing how to combine them with learning-based methods.

More Challenging Open World Assumption. In the inductive setting, the entity set $\mathcal{E}$ is divided into an observed set $\mathcal{E}_o^r$ and an inempty inductive set $\mathcal{E}_i^r$ with respect to the relation $r \in \mathcal{R}$, that is $\mathcal{E}_o^r \cap \mathcal{E}_i^r = \emptyset, \mathcal{E}_o^r \cup \mathcal{E}_i^r = \mathcal{E}, \forall r \in \mathcal{R}$. Under such a setting, the open world assumption in Definition 3 is rewritten as $I_{\mathcal{KG}_o}(r) \subseteq \mathcal{E}_o^r \times \mathcal{E}_o^r \subseteq I_{\mathcal{KG}_u}(r) \subseteq \mathcal{E} \times \mathcal{E}$. This setting makes it very hard to estimate the relations over $\mathcal{E} \times \mathcal{E} - \mathcal{E}_o^r \times \mathcal{E}_o^r$.

5.2 Towards First Order Queries and Beyond

The semantics of the logic queries lacks serious discussion. There is no standard dataset and benchmark for first-order queries with cycles, multiple free variables, or universal quantifiers even for the simplest L_{KG} language. Moreover, it is also worth discussing first-order queries over general relational data such as KGs with attributes and attribute functions. It is also worthwhile to expand the first-order logic to fixed-point or monadic second-order logic [30].

5.3 Towards Versatile Learning-based Models

Existing methods are incapable of addressing more and more challenging queries and complex data discussed above. It is essential to investigate the graph or sequence-based query embedding models to solve various queries. Given the combinatorial natures of the query answering problems, new opportunities could be found with neural combinatorial solvers [7]. Meanwhile, large language models, as a proven effective way to encapsulate world knowledge [39, 10], could be considered as another approach to modeling the relational data.

6 Conclusion

Big data and world knowledge are usually uncertain and dynamic. Logical queries over knowledge graphs are essential ways to understand the world as a simplified data model through the lens of logic and reasoning. In this paper, we structure the existing knowledge graph logical query answering tasks and methods as a three-party

game between data, query, and models. With the rigorous concepts from finite model theory [36, 30], our work portrays a detailed landscape of the achievements and limitations of existing work. Our discussion shows that there are various gaps between current research to the ambitious goal in terms of data, queries, and models. We hope our paper can be a practical guide for traveling in the uncertain and dynamic data world.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. Foundations of Databases. Addison-Wesley, Reading, Mass, 1995.

[2] D. Alivanistos, M. Berrendorf, M. Cochez, and M. Galkin. Query Embedding on Hyper-Relational Knowledge Graphs. In International Conference on Learning Representations, Mar. 2022.

[3] A. Amayuelas, S. Zhang, S. X. Rao, and C. Zhang. Neural Methods for Logical Reasoning over Knowledge Graphs. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022.

[4] E. Arakelyan, D. Daza, P. Minervini, and M. Cochez. Complex Query Answering with Neural Link Predictors. In International Conference on Learning Representations, 2021.

[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, editors, The Semantic Web, Lecture Notes in Computer Science, pages 722–735, Berlin, Heidelberg, 2007. Springer.

[6] J. Bai, Z. Wang, H. Zhang, and Y. Song. Query2Particles: Knowledge Graph Reasoning with Particle Embeddings. In Findings of the Association for Computational Linguistics: NAACL 2022, pages 2703–2714, Seattle, United States, July 2022. Association for Computational Linguistics.

[7] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings. OpenReview.net, 2017.

[8] K. D. Bollacker, C. Evans, P. K. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In J. T.-L. Wang, editor, Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008, pages 1247–1250. ACM, 2008.

[9] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013.

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

[11] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr, and T. M. Mitchell. Toward an Architecture for Never-Ending Language Learning. In M. Fox and D. Poole, editors, Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010. AAAI Press, 2010.

[12] X. Chen, Z. Hu, and Y. Sun. Fuzzy Logic Based Logical Query Answering on Knowledge Graphs. In Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022, pages 3939–3948. AAAI Press, 2022.

[13] N. Choudhary, N. Rao, S. Katariya, K. Subbian, and C. Reddy. Probabilistic Entity Representation Model for Reasoning over Knowledge Graphs. In Advances in Neural Information Processing Systems, volume 34, pages 23440–23451. Curran Associates, Inc., 2021.

[14] N. Choudhary, N. Rao, S. Katariya, K. Subbian, and C. K. Reddy. Self-Supervised Hyperboloid Representations from Logical Queries over Knowledge Graphs. In Proceedings of the Web Conference 2021, WWW '21, pages 1373–1384, New York, NY, USA, June 2021. Association for Computing Machinery.

[15] W. Cohen, F. Yang, and K. R. Mazaitis. TensorLog: A Probabilistic Database Implemented Using Deep-Learning Infrastructure. Journal of Artificial Intelligence Research, 67:285–325, Feb. 2020.

[16] D. Daza and M. Cochez. Message Passing Query Embedding, June 2020.

[17] E. A. Emerson. Temporal and modal logic. In Formal Models and Semantics, pages 995–1072. Elsevier, 1990.

[18] M. Galkin, P. Trivedi, G. Maheshwari, R. Usbeck, and J. Lehmann. Message Passing for Hyper-Relational Knowledge Graphs. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 7346–7359, Online, Nov. 2020. Association for Computational Linguistics.

[19] M. Galkin, Z. Zhu, H. Ren, and J. Tang. Inductive Logical Query Answering in Knowledge Graphs. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2022 NeurIPS 2022, November 27- December 9, 2022, New Orleans, page 25, 2022.

[20] P. Hájek. Metamathematics of Fuzzy Logic, volume 4 of Trends in Logic. Springer Netherlands, Dordrecht, 1998.

[21] W. L. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky, and J. Leskovec. Embedding Logical Queries on Knowledge Graphs. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pages 2030–2041, 2018.

[22] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In Advances in Neural Information Processing Systems, volume 33, pages 22118–22133. Curran Associates, Inc., 2020.

[23] Z. Hu, V. Gutierrez Basulto, Z. Xiang, X. Li, R. Li, and J. Z. Pan. Type-aware Embeddings for Multi-Hop Reasoning over Knowledge Graphs. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, pages 3078–3084, Vienna, Austria, July 2022. International Joint Conferences on Artificial Intelligence Organization.

[24] Z. Huang, M.-F. Chiang, and W.-C. Lee. LinE: Logical Query Reasoning over Hierarchical Knowledge Graphs. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, pages 615–625, New York, NY, USA, Aug. 2022. Association for Computing Machinery.

[25] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems, 33(2):494–514, Feb. 2022.

[26] Z. Jia, S. Pramanik, R. S. Roy, and G. Weikum. Complex Temporal Question Answering on Knowledge Graphs. In G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, and H. Tong, editors, CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021, pages 792–802. ACM, 2021.

[27] A. K. Katsaras and D. B. Liu. Fuzzy vector spaces and fuzzy topological vector spaces. Journal of Mathematical Analysis and Applications, 58(1):135–146, Mar. 1977.

[28] B. Kotnis, C. Lawrence, and M. Niepert. Answering Complex Queries in Knowledge Graphs with Bidirectional Sequence Encoders. Proceedings of the AAAI Conference on Artificial Intelligence, 35(6):4968–4977, May 2021.

[29] D. M. Kroenke, D. J. Auer, S. L. Vandenberg, and R. C. Yoder. Database Processing: Fundamentals, Design, and Implementation. Pearson, NY NY, 15th edition, 40th anniversary edition edition, 2018.

[30] L. Libkin. Elements of Finite Model Theory. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[31] L. Libkin and C. Sirangelo. Open and Closed World Assumptions in Data Exchange. In B. C. Grau, I. Horrocks, B. Motik, and U. Sattler, editors, Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30, 2009, volume 477 of CEUR Workshop Proceedings. CEUR-WS.org, 2009.

[32] B. Y. Lin, X. Chen, J. Chen, and X. Ren. KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning. EMNLP/IJCNLP, 2019.

[33] L. Liu, B. Du, H. Ji, C. Zhai, and H. Tong. Neural-Answering Logical Queries on Knowledge Graphs. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21, pages 1087–1097, New York, NY, USA, Aug. 2021. Association for Computing Machinery.

[34] X. Liu, S. Zhao, K. Su, Y. Cen, J. Qiu, M. Zhang, W. Wu, Y. Dong, and J. Tang. Mask and Reason: Pre-Training Knowledge Graph Transformers for Complex Logical Queries. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, pages 1120–1130, New York, NY, USA, Aug. 2022. Association for Computing Machinery.

[35] F. Luus, P. Sen, P. Kapanipathi, R. Riegel, N. Makondo, T. Lebese, and A. Gray. Logic Embeddings for Complex Query Answering. arXiv:2103.00418 [cs], Feb. 2021.

[36] D. Marker. Model Theory: An Introduction. Number 217 in Graduate Texts in Mathematics. Springer, New York, 2002.

[37] G. A. Miller. WordNet: A lexical database for English. Communications of the ACM, 38(11):39–41, Nov. 1995.

[38] T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner, and L. Pintscher. From Freebase to Wikidata: The Great Migration. In Proceedings of the 25th International Conference on World Wide Web, WWW '16, pages 1419–1428, Republic and Canton of Geneva, CHE, Apr. 2016. International World Wide Web Conferences Steering Committee.

[39] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller. Language models as knowledge bases? In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2463–2473, 2019.

[40] H. Ren, H. Dai, B. Dai, X. Chen, M. Yasunaga, H. Sun, D. Schuurmans, J. Leskovec, and D. Zhou. LEGO: Latent Execution-Guided Reasoning for Multi-Hop Question Answering on Knowledge Graphs. In Proceedings of the 38th International Conference on Machine Learning, pages 8959–8970. PMLR, July 2021.

[41] H. Ren, H. Dai, B. Dai, X. Chen, D. Zhou, J. Leskovec, and D. 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, KDD '22, pages 1472–1482, New York, NY, USA, Aug. 2022. Association for Computing Machinery.

[42] H. Ren, W. Hu, and J. Leskovec. Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020.

[43] H. Ren and J. Leskovec. Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. In H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H.-T. Lin, editors, Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual, 2020.

[44] D. Ruffinelli, S. Broscheit, and R. Gemulla. You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings. In International Conference on Learning Representations, Mar. 2020.

[45] A. Saxena, S. Chakrabarti, and P. P. Talukdar. Question Answering Over Temporal Knowledge Graphs. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pages 6663–6676. Association for Computational Linguistics, 2021.

[46] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling Relational Data with Graph Convolutional Networks. In A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, editors, The Semantic Web, Lecture Notes in Computer Science, pages 593–607, Cham, 2018. Springer International Publishing.

[47] Y. Sun, Q. Shi, L. Qi, and Y. Zhang. JointLK: Joint Reasoning with Language Models and Knowledge Graphs for Commonsense Question Answering. In M. Carpuat, M.-C. de Marneffe, and I. V. M. Ruíz, editors, Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022, pages 5049–5060. Association for Computational Linguistics, 2022.

[48] A. Talmor and J. Berant. The Web as a Knowledge-Base for Answering Complex Questions. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 641–651, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[49] K. Teru, E. Denis, and W. L. Hamilton. Inductive Relation Prediction by Subgraph Reasoning. In Proceedings of the 37th International Conference on Machine Learning, pages 9448–9457. PMLR, Nov. 2020.

[50] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. Advances in Neural Information Processing Systems, 34:24261–24272, 2021.

[51] K. Toutanova and D. Chen. Observed versus latent features for knowledge base and text inference. In Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality, pages 57–66, Beijing, China, July 2015. Association for Computational Linguistics.

[52] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard. Complex Embeddings for Simple Link Prediction. In Proceedings of The 33rd International Conference on Machine Learning, pages 2071–2080. PMLR, June 2016.

[53] G. Van den Broeck and D. Suciu. Query Processing on Probabilistic Data: A Survey. Foundations and Trends® in Databases, 7(3-4):197–341, 2017.

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

[55] Z. Wang, H. Yin, and Y. Song. Benchmarking the Combinatorial Generalizability of Complex Query Answering on Knowledge Graphs. In J. Vanschoren and S.-K. Yeung, editors, Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, Virtual, 2021.

[56] W. Xiong, T. Hoang, and W. Y. Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 564–573, 2017.

[57] Z. Xu, W. Zhang, P. Ye, H. Chen, and H. Chen. Neural-Symbolic Entangled Framework for Complex Query Answering. In Advances in Neural Information Processing Systems, Oct. 2022.

[58] D. Yang, P. Qing, Y. Li, H. Lu, and X. Lin. GammaE: Gamma Embeddings for Logical Queries on Knowledge Graphs, Oct. 2022.

[59] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021, pages 535–546. Association for Computational Linguistics, 2021.

[60] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep Sets. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[61] R. Zhang, D. Hristovski, D. Schutte, A. Kastrin, M. Fiszman, and H. Kilicoglu. Drug repurposing for COVID-19 via knowledge graph completion. Journal of Biomedical Informatics, 115:103696, Mar. 2021.

[62] W. Zhang, J. Chen, J. Li, Z. Xu, J. Z. Pan, and H. Chen. Knowledge Graph Reasoning with Logics and Embeddings: Survey and Perspective. arXiv:2202.07412 [cs], Feb. 2022.

[63] Y. Zhang, P. Li, H. Liang, A. Jatowt, and Z. Yang. Fact-Tree Reasoning for N-ary Question Answering over Knowledge Graphs. In S. Muresan, P. Nakov, and A. Villavicencio, editors, Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022, pages 788–802. Association for Computational Linguistics, 2022.

[64] Z. Zhang, J. Wang, J. Chen, S. Ji, and F. Wu. ConE: Cone Embeddings for Multi-Hop Reasoning over Knowledge Graphs. In Advances in Neural Information Processing Systems, volume 34, pages 19172–19183. Curran Associates, Inc., 2021.

[65] Z. Zhu, M. Galkin, Z. Zhang, and J. Tang. Neural-Symbolic Models for Logical Queries on Knowledge Graphs, May 2022.

[66] Z. Zhu, Z. Zhang, L.-P. Xhonneux, and J. Tang. Neural Bellman-Ford Networks: A General Graph Neural Network Framework for Link Prediction. In Advances in Neural Information Processing Systems, volume 34, pages 29476–29490. Curran Associates, Inc., 2021.

Knowledge Graph Comparative Reasoning for Fact Checking: Problem Definition and Algorithms

Lihui Liu*, Ruining Zhao*, Boxin Du‡,*, Yi Ren Fung*, Heng Ji*, Jiejun Xu†, Hanghang Tong*
*Department of Computer Science, University of Illinois at Urbana Champaign
†HRL Laboratories, LLC., jxu@hrl.com
‡Amazon, boxin@amazon.com
*lihuil2, ruining9, yifung2, hengji, htong@illinois.edu

Abstract

Knowledge graphs are ubiquitous data structure which have been used in many applications. Knowledge graph reasoning aims to discover or infer knowledge based on existing information in the knowledge graph. However, most of the existing works belong to point-wise approaches, which perform reasoning w.r.t. a single piece of clue. Comparative reasoning over knowledge graph focuses on inferring commonality and inconsistency with respect to multiple pieces of clues which is a new research direction and can be applied to many applications. In this paper, we formally give the definition of comparative reasoning and propose several different methods to tackle comparative reasoning in both pairwise and collective cases. The idea of the proposed methods is that we find a knowledge segment from the knowledge graph to best represent the semantic meaning of the given claim, and reasons according to it. We perform extensive empirical evaluations on real-world datasets to demonstrate that the proposed methods have good performances.

1 Introduction

Knowledge graphs are ubiquitous data structure which are used to store really world entities (e.g., `Alan Turing`) and their relations (`Alan Turing`, `wasBornIn`, `United Kingdom`). Since the debut in 2012, several widely used knowledge graphs have been proposed, which include Yago, Wikidata, Freebase and so on. Knowledge graph reasoning which aims to discover/explain existing knowledge or infer new knowledge from existing information in the knowledge graph has emerged as an important research direction over the last few years [20].

Despite the great achievement in both academia and industry, most of the existing works on knowledge graph reasoning belong to the point-wise approaches, which perform reasoning w.r.t. a single piece of clue (e.g., a triple [1], a multi-hop query [20], a complex query graph [17]). For example in fact checking, given a claim (e.g., represented as a triple of the knowledge graph), it decides whether the claim is authentic or falsified [26, 2]. However, comparative reasoning ([18, 19]) is rarely studied. Different from point-wise reasoning (or reasoning

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*The publication was written prior to the employee joining Amazon.

over knowledge graph). Comparative reasoning over knowledge graph [18] focuses on inferring commonality and/or inconsistency with respect to multiple pieces of clues (e.g., multiple claims about a news article), which is a new research direction over knowledge graphs and can be widely applied to other applications, e.g., fact checking.

Comparative reasoning has many unique advantages compared with point-wise (single claim) fact checking. This is because in many real-world situations, e.g., multimodal fake news detection [21], single claim fact checking alone is insufficient, while comparative reasoning offers a more complete picture w.r.t. the input clues, which in turn helps the users discover the subtle patterns (e.g., inconsistency) that would be invisible by point-wise approaches. When we verify the two claims/triples at the same time, the result may be inconsistent even though each claim/triple itself is consistent if we evaluate it individually. Figure 1 gives an example to illustrate the power of comparative reasoning. Suppose there is a multi-modal news article and we wish to verify its truthfulness. To this end, two query graphs are extracted from the given news, respectively. One query graph contains all the information from the text, and the other contains the information from the image. If we perform point-wise reasoning to check each of these two query graphs separately, both seem to be true. However, if we perform reasoning w.r.t. both query graphs simultaneously, and by comparison, we could discover the subtle inconsistency between them (i.e., the different air plane types, the difference in maximum flying distances). In addition, comparative reasoning can also be used in knowledge graph expansion, integration and completion [18].

In this paper, we address the problem of comparative reasoning. We mainly focus on two problems: pairwise comparative reasoning and collective comparative reasoning. To be specific, we address two key challenges as follows. We leverage graph neural network and graph kernel to reveal the commonality and inconsistency among input clues according to the information in the background knowledge graph. We propose several different algorithms and demonstrate their effectiveness. A common building block of comparative reasoning is knowledge segment, which is a small connection subgraph of a given clue (e.g., a triple or part of it) to summarize its semantic context. Based on that, we present core algorithms to enable both pairwise reasoning and collective reasoning. The key idea is to use the structure and semantic information in knowledge segments to help discover vague contradictions.

The main contributions of the paper are

- **Problem Definition.** We introduce comparative reasoning over knowledge graphs, which complements and expands the existing point-wise reasoning capabilities.
- **Algorithms.** We propose a family of comparative reasoning algorithms which can solve both pairwise comparative reasoning and collective comparative reasoning.
- **Empirical Evaluations.** We perform extensive empirical evaluations to demonstrate the efficacy of our proposed methods.

The rest of the paper is organized as follows. Section 2 introduces notations used in this paper and gives the problem definition. Section 3 introduces how to extract the knowledge segment from the knowledge graph. Section 4 proposes different methods to solve comparative reasoning problem. The experiment results are presented in Section 5, and the related work is reviewed in Section 6. Finally, the paper is concluded in Section 7.

2 Problem Definition

In this section, we first introduce the symbols that will be used throughout the paper, then we introduce other important concepts and formally define the comparative reasoning problem.

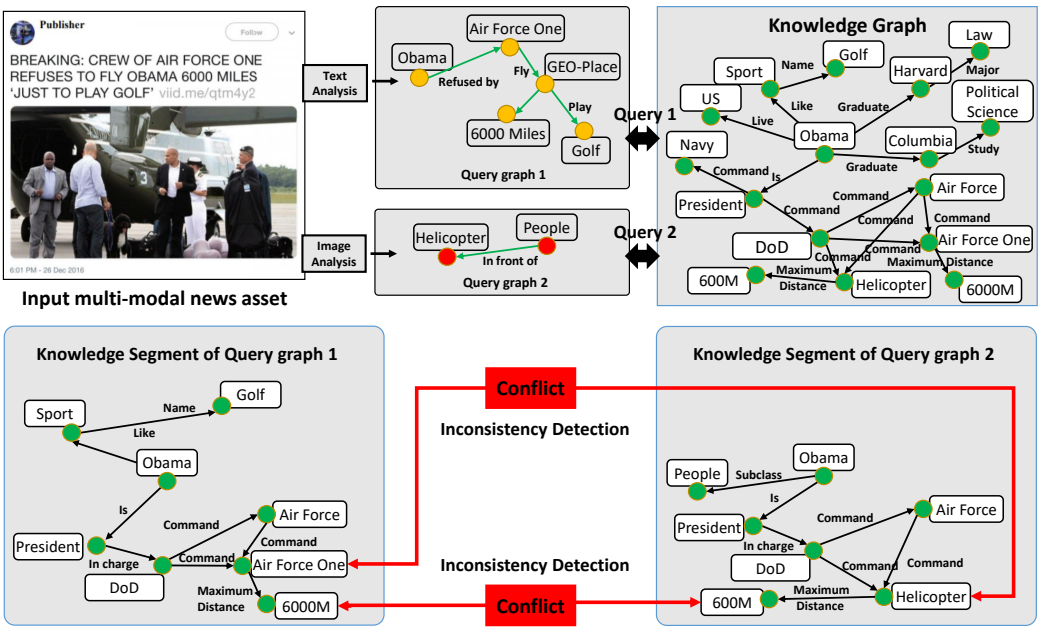


Figure 1: An illustrative example of using comparative reasoning for semantic inconsistency detection. Source of the image at the top-left: [4]. The example is borrowed from [18].

Table 4: Notations and definitions

Symbols	Definition	Symbols	Definition
$\mathcal{G}$=$\{V^G, E^G, L^G\}$	a knowledge graph	v_i	the i^{th} entity/node in knowledge graph
r_i	the i^{th} relation/edge in knowledge graph	e_i	the i^{th} given by the user
Q=$\{V^Q, E^Q, L^Q\}$	an attributed query graph	KS_i	knowledge segment i
A_i	adjacency matrix of KS_i	$A_\times$	kronecker product of A_1 and A_2
N^l	diagonal matrix of the l^{th} node attribute	$N_\times$	combined node attribute matrix
N_i	attribute matrix of KS_i, the j^{th} row denotes the attribute vector of node j in KS_i	$S^{i,j}$	single entry matrix $S^{i,j}(i,j) = 1$ and zeros elsewhere

Table 4 gives the main notations used throughout this paper. A knowledge graph can be denoted as $\mathcal{G} = (V, R, E)$ where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes/entities, $R = \{r_1, r_2, ..., r_m\}$ is the set of relations and E is the set of triples. Each triple in the knowledge graph can be denoted as (h, r, t) where $h \in V$ is the head (i.e., subject) of the triple, $t \in V$ is the tail (i.e., object) of the triple and $r \in R$ is the edge (i.e., relation, predicate) of the triple which connects the head h to the tail t.

Given multiple pieces of clues, the goal of comparative reasoning is to infer commonality and/or inconsistency among them. If the given information is a pair of clues, we call it pairwise comparative reasoning or pairwise fact checking. The goal is to deduce whether these two clues are coherent or not. If the given information is a connected query graph, the goal is to detect whether there is inconsistency inside the given graph. The problem is called collective comparative reasoning or collective fact checking. Different from traditional point-wise reasoning methods, comparative reasoning can unveil some subtle patterns which point-wise approaches may overlook. Take knowledge graph based fact checking as an example, considering two claims/triples: (`Barack Obama`, `graduatedFrom`, `Harvard University`) and (`Barack Obama`, `majorIn`, `Political Science`). Even each clue/claim is true, but if we check them together at the same time, we can see that they cannot be both true. This is because `Barack Obama` majored in `law` instead of `Political Science` when he studied at `Harvard University`. So, we might fail to detect the inconsistency between them without appropriately examining different clues/claims together.

To facilitate comparative reasoning, how to utilize the background information in knowledge graph is an important problem. If we can find a subgraph in the knowledge graph which can best express the semantic meaning of each input clue, the hidden conflicts can be easier to detect. Ideally, this subgraph should contain all the meaningful/important entities and relations in the knowledge graph which are related to the given clue. We call this subgraph knowledge segment, which is formally defined as follows.

Definition 1: Knowledge Segment (KS for short) is a connection subgraph of the knowledge graph that best describes the semantic context of a piece of given clue (i.e., a node, a triple or a query graph).

Figure 1 gives an example of knowledge segment. Given two clues which are two query graphs extracted from the text and image respective, their corresponding knowledge segments are shown in the bottom of the Figure. As we can see, expressing the given clues with knowledge segments can help us detect the inconsistency without difficulty.

Given the knowledge segments of multiple pieces of clues, comparative reasoning aims to infer the commonality and inconsistency among these knowledge segments to make decision. For pairwise case, the commonality refers to the common elements of these two knowledge segments. The inconsistency includes any elements that are contradicts with each other. Assuming the two given clues are two edges/triples: $E_1^Q =< \mathtt{s}_1, \mathtt{p}_1, \mathtt{o}_1 >$ and $E_2^Q =< \mathtt{s}_2, \mathtt{p}_2, \mathtt{o}_2 >$ where $\mathtt{s}_1, \mathtt{o}_1, \mathtt{s}_2, \mathtt{o}_2 \in V$ and $\mathtt{p}_1, \mathtt{p}_2 \in E$. We denote their corresponding knowledge segments as KS_1 for E_1^Q and KS_2 for E_2^Q, respectively. The commonality and inconsistency between these two knowledge segments are defined as follows.

Definition 2: Commonality. Given two triples (E_1^Q and E_2^Q) and their knowledge segments (KS_1 and KS_2), the commonality of these two triples refers to the shared nodes and edges between E_1^Q and E_2^Q, as well as the shared nodes and edges between KS_1 and KS_2: $((V^{KS_1} \cap V^{KS_2}) \cup (V^{Q_1} \cap V^{Q_2}), (E^{KS_1} \cap E^{KS_2}) \cup (E^{Q_1} \cap E^{Q_2}))$.

Definition 3: Inconsistency. Given two knowledge segments KS_1 and KS_2, the inconsistency between these two knowledge segments refers to any element (node, node attribute or edge) in KS_1 and KS_2 that contradicts with each other.

Different from pairwise comparative reasoning, collective comparative reasoning aims to find the commonality and/or inconsistency inside a query graph which consists of a set of inter-connected edges/triples. The corresponding definition is given below.

Definition 4: Collective Commonality. For each edge E_i^Q in a query graph Q, let KS_i be its knowledge segment. The collective commonality between any triple pair in the query graph is the intersection of their knowledge segments.

Definition 5: Collective Inconsistency. For each edge E_i^Q in a query graph Q, let KS_i be its knowledge segment. The collective inconsistency refers to any elements (node or edge or node attribute) in these knowledge segments that contradict with each other.

Given the above notation and information, the problem of comparative reasoning is formal defined as:

Problem Definition 1: Pairwise Comparative Reasoning:
Given: (1) A knowledge graph $\mathcal{G}$, (2) two triples E_1^Q and E_2^Q;
Output: A binary decision regarding the consistency of E_1^Q and E_2^Q.

Problem Definition 2: Collective Comparative Reasoning:
Given: (1) A knowledge graph $\mathcal{G}$, (2) a query graph Q;
Output: A binary decision regarding the consistency of Q.

3 Knowledge Segment Extraction

In this section, we introduce how to extra knowledge segment to best express the semantic meaning of a given claim. We first introduce how to transform the knowledge graph into a relation specified weighted graph, then introduce how to extract Edge-specific Knowledge Segment and Subgraph-specific Knowledge Segment from it.

Generally speaking, given a clue (e.g., a triple or a query graph) from the user, the goal of knowledge segment extraction is to extra a subgraph which can best express the semantic meaning of the given clue. Many existing methods have been proposed to extract a concise subgraph from the source node of the querying edge to its target node in weighted or unweighted graphs. For example, multi-hop method [9], minimum cost maximum flow method [24], K-simple shortest paths based method [8] or connection subgraph [7], [11], [23] extraction methods.

However, these methods do not directly apply to knowledge graphs because the edges (i.e., predicates) of a knowledge graph have specific semantic meanings (e.g., types, relations). To address this issue, we seek to convert the knowledge graph to a weighted graph by designing a predicate-predicate similarity measure for knowledge segment extraction.

3.1 Predicate-Predicate Similarity

In order to transform the knowledge graph into weighted graph, We propose to use a TF-IDF based method to measure the similarity between different predicates, and transfer the knowledge graph into a weighted graph whose edge weight represents the similarity between the edge predicate and query predicate. The key idea behind TF-IDF based method is that we can treat each triple in the knowledge graph and its adjacent neighboring triples as a document, and use a TF-IDF like weighting strategy to calculate the predicate similarity. For example, predicate `receiveDegreeFrom` may have neighbor predicates `major` and `graduateFrom`. These predicates should have high similarity with each other.

To be specific, we use the knowledge graph to build a co-occurrence matrix of predicates, and calculate their similarity by a TF-IDF like weighting strategy as follows. Let i, j denote two different predicates. We define the TF between two predicates as $\text{TF}(i,j) = \log(1 + C(i,j)w(j))$, where $C(i,j)$ is the co-occurrence of predicate i and j. The IDF is defined as $\text{IDF}(j) = \log \frac{|M|}{|\{i:C(i,j)>0\}|}$, where M is the number of predicates in the knowledge graph. Then, we build a TF-IDF weighted co-occurrence matrix U as $U(i,j) = \text{TF}(i,j) \times IDF(j)$. Finally, the similarity of two predicates is defined as $\text{Sim(i, j)} = \text{Cosine}(U_i, U_j)$, where U_i and U_j are the i^{th} row and j^{th} row of U, respectively.

For the predicate-predicate similarity, suppose we want to calculate the similarity between `major` and `study`. Both `major` and `study` have only one adjacent neighboring predicate `graduate`. This means that for any predicate $i \neq$ `graduate`, $U(\texttt{major}, i) = U(\texttt{study}, i) = 0$. Since $\text{E}(\texttt{graduate}) = 0$, we have $w(\texttt{graduate}) = 2\sigma(\infty) - 1 = 1$. We have $\text{TF}(\texttt{major}, \texttt{graduate}) = \text{TF}(\texttt{study}, \texttt{graduate}) = \log(1 + 1 \times 1) = 1$, and $U(\texttt{major}, \texttt{graduate}) = U(\texttt{study}, \texttt{graduate}) = IDF(\texttt{graduate}) = \log \frac{8}{4} = 1$. If we compare the two vectors, $U_{\texttt{major}}$ and $U_{\texttt{study}}$, we find that they are the same. Therefore, we have that $\text{Sim}(\texttt{major}, \texttt{study}) = 1$.

3.2 Edge-specific Knowledge Segment

Edge-specific knowledge segment extraction aims at finding a knowledge segment to best characterize the semantic context of the given edge (i.e., a triple). Several connection subgraph extraction methods exist for a weighted graph, e.g., [33] uses a random walk with restart based method to find an approximate subgraph; [11] uses maximal network flow to find a subgraph and [8] aims to find a denser local graph partitions. In this paper, after transforming the knowledge graph into a weighted graph, we find k-simple shortest paths [11] from the subject to the object of the given query edge as its knowledge segment.

3.3 Subgraph-specific Knowledge Segment

Following the idea of edge-specific knowledge segment extraction, we extract a knowledge segment for each edge in the given subgraph and we call the graph which contains all the edge-specific knowledge segments subgraph-specific knowledge segment. In other words, a subgraph-specific knowledge segment consists of multiple inter-linked edge-specific knowledge segments (i.e., one edge-specific knowledge segment for each edge of the input query subgraph).

The subgraph-specific knowledge segment provides richer semantics, including both the semantics for each edge of the query graph and the semantics for the relationship between different edges of the input query graph.

4 Comparative Reasoning

In this section, we introduce the technical details behind comparative reasoning. We first introduce on what condition we need to exert pairwise reasoning for two pieces of clues (e.g., two edges/triples), and then introduce two methods which focus on pairwise reasoning. Finally, we present the collective comparative reasoning. The main idea behind these functions is that we use a knowledge segment to express the semantic meaning of each query triple, and check the inconsistency according to information in the knowledge segments.

4.1 Pairwise Comparative Reasoning Condition

Given a pair of clues $< \mathtt{s}_1, \mathtt{p}_1, \mathtt{o}_1 >$ and $< \mathtt{s}_2, \mathtt{p}_2, \mathtt{o}_2 >$, we can divide it into the following six cases, including

C1. $\mathtt{s}_1 \neq \mathtt{s}_2, \mathtt{s}_1 \neq \mathtt{o}_2, \mathtt{o}_1 \neq \mathtt{s}_2, \mathtt{o}_1 \neq \mathtt{o}_2$. For this case, these two clues apparently refer to different things, e.g., <Alan Turing, wasBornIn, United Kingdom> and <Google, isLocatedIn, USA>.

C2. $\mathtt{s}_1 = \mathtt{s}_2$ and $\mathtt{o}_1 = \mathtt{o}_2$. If $\mathtt{p}_1 = \mathtt{p}_2$, these two clues are the same. If $\mathtt{p}_1$ and $\mathtt{p}_2$ are different or irrelevant, e.g., $\mathtt{p}_1 =$ wasBornIn, $\mathtt{p}_2 =$ hasWebsite, these two clues refer to different things. However, if $\mathtt{p}_1$ contradicts $\mathtt{p}_2$, they are inconsistent with each other.

C3. $\mathtt{s}_1 = \mathtt{s}_2$ but $\mathtt{p}_1 \neq \mathtt{p}_2$ and $\mathtt{o}_1 \neq \mathtt{o}_2$, e.g., <Alan Turing, wasBornIn, Maida Vale>, <Alan Turing, livesIn, United Kingdom>.

C4. $\mathtt{s}_1 = \mathtt{s}_2$, $\mathtt{p}_1 = \mathtt{p}_2$, but $\mathtt{o}_1 \neq \mathtt{o}_2$, e.g., <Alan Turing, wasBornIn, Maida Vale>, <Alan Turing, wasBornIn, United Kingdom>.

C5. $\mathtt{o}_1 = \mathtt{o}_2$, but $\mathtt{s}_1 \neq \mathtt{s}_2$. For this case, no matter what $\mathtt{p}_1$ and $\mathtt{p}_2$ are, these two clues refer to different things.

C6. $\mathtt{o}_1 = \mathtt{s}_2$. For this case, no matter what $\mathtt{p}_1$ and $\mathtt{p}_2$ are, they refer to different things. For example, <Alan Turing, wasBornIn, United Kingdom>, <United Kingdom, dealsWith, USA>.

Among these six cases, we can see that the clue pair in C1, C5 and C6 refer to different things. Therefore, there is no need to check the inconsistency between them. For C2, we only need to check the semantic meaning of their predicates, i.e., whether $\mathtt{p}_1$ contradicts $\mathtt{p}_2$. For example, $\mathtt{p}_1 =$ isFather and $\mathtt{p}_2 =$ isSon, they are inconsistent with each other. Otherwise, there is no inconsistency between them. We mainly focus on C3 and C4 where the two clues may be inconsistent with each other even if each of them is true. For example, either <Barack Obama, graduatedFrom, Harvard University> or <Barack Obama, majorIn , Political Science> could be true. But putting them together, they cannot be both true, since Barack Obama majored in law instead of Political Science when he studied at Harvard University. In other words, they are mutually exclusive with each other and thus are inconsistent. However, queries like <Alan Turing, wasBornIn, Maida Vale> and <Alan Turing, wasBornIn, United

Kingdom> are both true, because Maida Vale belongs to United Kingdom. Alternatively, we can say that United Kingdom contains Maida Vale. We summarize that if (1) the subjects of two clues are the same, and (2) their predicates are similar with each other or the same, they refer to the same thing. Furthermore, if their objects are two uncorrelated entities, it is highly likely that these two clues are inconsistent with each other.

Based on the above observations, we take the following three steps for pairwise comparative reasoning. First, given a pair of clues, we decide which of six cases it belongs to, by checking the subjects, predicates and objects of these two clues. Second, if this pair of clues belongs to C3 or C4, we need to further decide whether they are consistent with each other. In the following sections, we illustrate how to tackle this case.

4.2 Neural Network Based Pairwise Comparative Reasoning

Given two knowledge segments of a pair of clues which belong to C3 or C4, we treat each knowledge segment as an attributed graph, and adopt some ideas from network alignment to facilitate comparative reasoning. The basic idea is that if the two knowledge segments are consistent, most of their nodes must be able to align with or close to each other in the embedding space. Otherwise, the embedding distance of the inconsistent nodes should be large. Generally, the inconsistent checking problem is similar to anomaly detection or dissimilarity detection problem in the embedding space.

When reasoning a pair of knowledge segments, we consider two kinds of information: structure information and semantic information. We envision that both of them are important. For example, in Figure 1, Air Force One and Helicopter have similar structure information because they have many common neighbors, but their semantic meanings are very different, this may indicate a potential inconsistency between the two knowledge segments. On the other hand, although Air Force One and Helicopter have different structure information (when considering edge type), they also have different semantic information. This prompts that they refer to the different things. Inspired by this observation, we propose a neural network model which considers both the structure information and semantic information of knowledge segments to achieve pairwise comparative reasoning.

To encode the structure similarity, we use random walk with restart (considering edge type) to encode the knowledge segment structure information. The similar idea has been used by many other works, e.g. [34], [38]. Given a set of anchor nodes, random walk with restart will calculate a score for each node in the knowledge segment w.r.t. each anchor node. If two nodes have the similar random walk with restart score vector, their structure similarity should be high. To encode the semantic information of the knowledge segment, we borrow some ideas from natural language processing. More specifically, we sample some paths from the knowledge graph, and treat each path as a sentence, nodes in the knowledge graph can be treated as words in the sentence. If two nodes occur in the same sentence, their semantic information should be similar.

4.2.1 Structure Embedding

Given two knowledge segments, the common nodes in these two knowledge segments can be treated as anchor entities, structure embedding aims to embed the nodes in the knowledge segments to a high dimension space while keeping their structure information. The key intuition is that, the set of anchor links $\mathcal{L}$ provides the landmarks for all nodes in both networks. Relative positions based on anchor links can form a unified space for all nodes regardless which network they belong to [38]. Therefore, we can use random walk with restart to measure the relative position between nodes and anchor links. Let KS_1 and KS_2 be the two knowledge segments. Given an anchor link $l \in \mathcal{L}$ (we use l_1 and l_2 to denote the linked entities in KS_1 and KS_2), the RWR score vector r_{l_1} of size $n_1 \times 1$ can be obtained

$$r_{l_1} = (1-\beta)\hat{W}_1 r_{l_1} + \beta e_{l_1} \tag{4}$$

where n_1 is the number of nodes in KS_1, $\hat{W}_1$ is the row normalized adjacency matrix of KS_1, β is the restart probability and e_{l_1} is one-hot vector with $e_{l_1}(l_1) = 1$ and all other entries are 0. We can solve the equation and

get the final expression of r_{l_1} as:

$$r_{l_1} = \beta(I - (1-\beta)\hat{W_1}^{-1}e_{l_1} \tag{5}$$

Note that if KS_1 and KS_2 are the same, they will have the same random walk with restart score matrices.

After we get the random walk with restart matrices, we then use a share neural network to learn the embedding of these two knowledge segments. In this way, the structure information of each node can be kept in the embedding space. The learned embedding for KS_1 can be calculated as:

$$E_1 = \text{NeuralNetwork}(\text{RWR}_1) \tag{6}$$

where RWR_1 is the random walk with restart score matrix of KS_1. RWR_2 can be calculated in the same way.

4.2.2 Semantic Embedding

To captivate the semantic meaning of each node in the knowledge segment, we randomly sample some paths in the background knowledge graph and treat each path as an utterance while each node as a word in the sentence. Then, we use a popular language model Bert [5] to learn the semantic embedding of each node. If two nodes occur at the same path, their semantic embedding should be close to each other, otherwise, their semantic embedding should be far from each other.

We concatenate the node semantic embedding and structure embedding of two knowledge segments and use graph neural network to learn the graph embedding of KS_1 and KS_2 respectively. Finally, we predict whether they are consistent with each other according to the graph embedding. The architecture of the algorithm is shown in Figure 2.

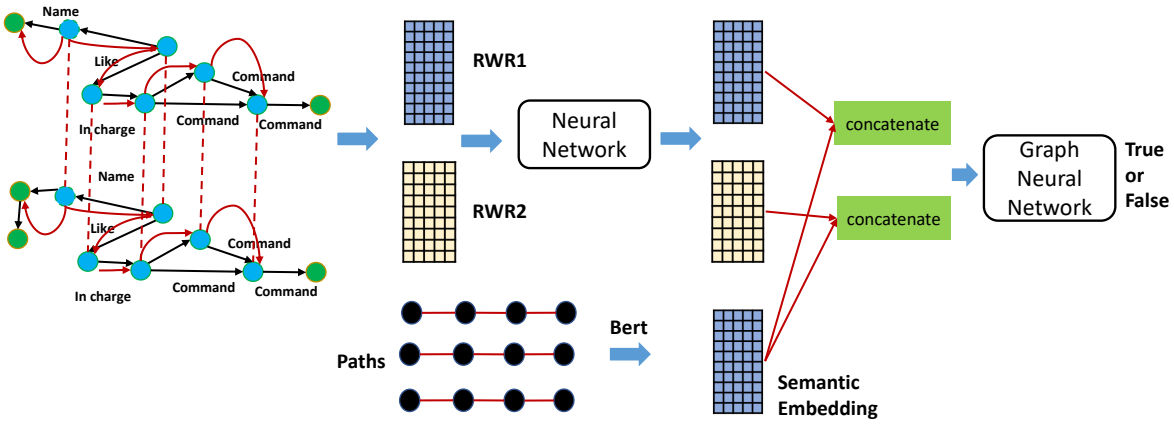


Figure 2: Neural network based pairwise comparative reasoning framework.

The pseudo code is given in Algorithm 1. GNN means graph neural network which is defined as follows

$$x^l_{N(u)} = \text{AGGREGATE}(\{\mathbf{X}^l(v,:), for v \in N(u)\}) \tag{7}$$

$$\mathbf{X}^{l+1}(u,:) = \sigma(\text{CONCAT}(\mathbf{X}^l(u,:), x^l_{N(u)})) \tag{8}$$

where AGGREGATE is the aggregation function, $\mathbf{X}$ is the input node embedding, l is the graph neural network layer number, and (u, v) are nodes in the input graph. The classifier in Algorithm 1 can be any machine learning model, e.g., SVM, logistic regression, decision tree and so on.

Algorithm 1 Neural Network Based Pairwise Comparative Reasoning

Input: Knowledge segment KS_1 and knowledge segment KS_2, pretrained embedding of all entities $\mathbf{E}$ in the knowledge graph.
Training:
Calculate random walks with restart matrices RWR_1 and RWR_2 for KS_1 and KS_2, respectively.
E_1 = NeuralNetwork(RWR_1)
E_2 = NeuralNetwork(RWR_2)
Get all node embedding in KS_1: $N_1 = \mathbf{E}(KS_1)$
Get all node embedding in KS_2: $N_2 = \mathbf{E}(KS_2)$
Concatenate embedding $C_1 = (E_1|N_1)$
Concatenate embedding $C_2 = (E_2|N_2)$
Predict result: Classifier(GNN(C_1), GNN(C_2))

4.3 Graph Kernel Based Pairwise Comparative Reasoning

Different from neural network based pairwise comparative reasoning, graph kernel based pairwise comparative reasoning aims to utilize graph kernel to find a set of key elements (nodes or edges or node attributes) in these two knowledge segments and then make decision according to these elements and related information in the knowledge graph. The idea is that if most of these key elements belong to the commonality of these two knowledge segments, it is highly likely that they refer to the same thing. Otherwise, these two clues refer to different things. Third, if they refer to the same thing, we further decide whether they conflict with each other. Here, the key idea is as follows. We build two new query triples $<$ `o1`, `isTypeOf`, `o2` $>$ and $<$ `o2`, `isTypeOf`, `o1` $>$. If one of them is true, the original two triples are consistent with each other. Otherwise, they are inconsistent.

In order to find the key elements, we propose to use the influence function w.r.t. the knowledge segment similarity [41]. The basic idea is that if we perturb a key element (e.g., change the attribute of a node or remove a node/edge), it would have a significant impact on the overall similarity between these two knowledge segments. Let KS_1 and KS_2 be the two knowledge segments. We can treat the knowledge segment as an attributed graph, where different entities have different attributes. We use random walk graph kernel with node attribute to measure the similarity between these two knowledge segments [41] [6].

$$\text{Sim}(KS_1, KS_2) = q'_{\times}(I - cN_{\times}A_{\times})^{-1}N_{\times}p_{\times} \tag{9}$$

where $q'_{\times}$ and $p_{\times}$ are the stopping probability distribution and the initial probability distribution of random walks on the product matrix, respectively. $N_{\times}$ is the combined node attribute matrix of the two knowledge segments $N_{\times} = \sum_{j=1}^{d} N_1^j \otimes N_2^j$ where N_i^j ($i \in \{1, 2\}$) is the diagonal matrix of the j^{th} column of attribute matrix N_i. $A_{\times}$ is the Kronecker product of the adjacency matrices of knowledge segments A_1 and A_2. $0 < c < 1$ is a parameter.

We propose to use the influence function $\text{Sim}(KS_1, KS_2)$ w.r.t. knowledge segment elements $\frac{\partial Sim(KS_1, KS_2)}{\partial e}$, where e represents an element of the knowledge segment KS_1 or KS_2. The element with a high absolute influence function value is treated as a key element, and it can be a node, an edge, or a node attribute. The influence function of different elements can be computed according to the following lemma. Note that the influence function w.r.t. elements in KS_2 can be computed in a similar way.

Lemma 6: (Knowledge Segment Similarity Influence Function [41].) Given $\text{Sim}(KS_1, KS_2)$ in Eq. (9). Let $Q = (I - cN_{\times}A_{\times})^{-1}$ and $S^{j,i}$ is a single entry matrix defined in Table 4. We have that

(i) The influence of an edge $A_1(i, j)$ in KS_1 can be calculated as

$$I(A_1(i,j)) = \frac{\partial \text{Sim}(KS_1, KS_2)}{\partial A_1(i,j)} = cq'_{\times}QN_{\times}[(S^{i,j} + S^{j,i}) \otimes A_2]QN_{\times}p_{\times} \tag{10}$$

(ii) The influence of a node i in KS_1 can be calculated as

$$I(N_1(i)) = \frac{\partial \text{Sim}(KS_1, KS_2)}{\partial N_1(i)} = cq'_{\times} QN_{\times}[\sum_{j|A_1(i,j)=1} (S^{i,j} + S^{j,i}) \otimes A_2]QN_{\times}p_{\times} \tag{11}$$

(iii) The influence of a node attribute j of node i in KS_1 can be calculated as

$$I(N_1^j(i,i)) = \frac{\partial \text{Sim}(KS_1, KS_2)}{\partial N_1^j(i,i)} = q'_{\times} Q[S^{i,i} \otimes N_2^j](I + cA_{\times}QN_{\times})p_{\times} \tag{12}$$

For a given knowledge segment, we flag the top 50% of the elements (e.g., node attribute, node and edge) with the highest absolute influence function values as key elements. We would like to check whether these key elements belong to the commonality of these two knowledge segments. If most of them (e.g., 60% or more) belong to the commonality of these two knowledge segments, we say the two query clues describe the same thing. Otherwise, they refer to different things and thus we do not need to check the inconsistency between them.

If we determine that the query clues refer to the same thing, the next step is to decide whether they are inconsistent with each other. That is, given query clues $<$ $\mathtt{s}_1$, $\mathtt{p}_1$, $\mathtt{o}_1$ $>$ and $<$ $\mathtt{s}_1$, $\mathtt{p}_2$, $\mathtt{o}_2$ $>$, we need to decide whether $\mathtt{o}_1$ belongs to $\mathtt{o}_2$ or $\mathtt{o}_2$ belongs to $\mathtt{o}_1$. To this end, we build two new queries $<$ $\mathtt{o}_1$, `isTypeOf`, $\mathtt{o}_2$ $>$ and $<$ $\mathtt{o}_2$, `isTypeOf`, $\mathtt{o}_1$ $>$. Then, we extract the knowledge segments for these two queries, and check whether these two segments are true. If one of them is true, we say the original clues are consistent with each other, otherwise they are inconsistent. After we extract the knowledge segments for $<$ $\mathtt{o}_1$, `isTypeOf`, $\mathtt{o}_2$ $>$ and $<$ $\mathtt{o}_2$, `isTypeOf`, $\mathtt{o}_1$ $>$, we treat each knowledge segment as a directed graph, and calculate how much information can be transferred from the subject to the object. We define the transferred information amount as:

$$\text{infTrans}(\mathtt{o}_1, \mathtt{o}_2) = \max_{1 \le j \le k} \text{pathValue}(j) \tag{13}$$

where pathValue(j) is defined as the multiplication of the weights in the path. For an edge, its weight is the predicate-predicate similarity Sim(`isTypeOf`, e_i). If $\max\{\text{infTrans}(\mathtt{o}_1, \mathtt{o}_2), \text{infTrans}(\mathtt{o}_2, \mathtt{o}_1)\}$ is larger than a threshold T, then we say $\mathtt{o}_1$ belongs to $\mathtt{o}_2$ or $\mathtt{o}_2$ belongs to $\mathtt{o}_1$. We set $T = 0.700$ in our experiment.

4.4 Graph Kernel Based Collective Comparative Reasoning

Different from pairwise comparative reasoning, collective comparative reasoning aims to find the commonality and/or inconsistency inside a query graph which consists of a set of inter-connected edges/triples. To check the inconsistency, one naive method is using the pairwise comparative reasoning method to check the inconsistency for each pair of edges in the query graph. However, this method is neither computationally efficient nor sufficient. For the former, if two clues (e.g., two claims from a news article) are weakly or not related with each other on the query graph, we might not need to check the inconsistency between them at all. For the latter, in some subtle situations, the semantic inconsistencies could only be identified when we collectively reason over multiple (more than two) knowledge segments. For example, given the following three claims, including (1) Obama is refused by Air Force One; (2) Obama is the president of the US; (3) The president of US is in front of a helicopter. Only if we reason these three claims collectively, can we identify the semantic inconsistency among them.

Based on the above observation, we propose the following method to detect the collective inconsistency.

First, we find a set of key elements inside the semantic matching subgraph. Different from pair-wise comparative reasoning, the importance/influence of an element for collective comparative reasoning is calculated by the entire semantic matching subgraph. More specifically, we first transform the query graph and its semantic matching subgraph (i.e., subgraph-specific knowledge segment) into two line graphs, which are defined as follows.

Definition 7: Line Graph [24]. For an arbitrary graph $G = (V, E)$, the line graph $L(G) = (V', E')$ of G has the following properties: (1) the node set of $L(G)$ is the edge set of G ($V' = E$); (2) two nodes V'_i, V'_j in $L(G)$ are adjacent if and only if the corresponding edges e_i, e_j of G are incident on the same node in G.

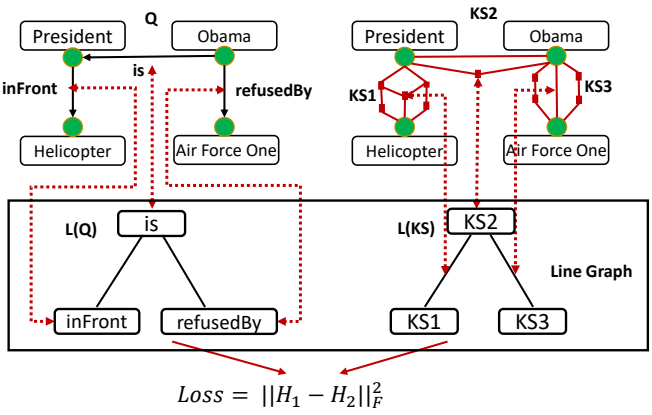


Figure 3: Collective comparative reasoning workflow.

Figure 3 gives an example of the line graph. For the line graph $L(Q)$, the edge weight is the predicate-predicate similarity of the two nodes it connects. For the line graph $L(KS)$, the edge weight is the knowledge segment similarity by Eq. (9) of the two nodes it connects. The rationality of building these two line graphs is that if the semantic matching subgraph is a good representation of the original query graph, the edge-edge similarity in $L(Q)$ would be similar to the knowledge segment similarity in $L(KS)$.

To measure the importance of an element, we propose to use the influence function w.r.t. the distance between $L(Q)$ and $L(KS)$. We assume that a key element, if perturbed, would have a great effect on the distance Loss $= ||H_1 - H_2||_F^2$, where H_1 is the weighted adjacency matrix of $L(Q)$, and H_2 is the weighted adjacency matrix of $L(KS)$. We use the influence function $\frac{\partial \text{Loss}(H_1,H_2)}{\partial e}$, where e represents an element of the knowledge segment graph and it could be a node, an edge, or a node attribute. Lemma 8 provides the details on how to compute such influence functions.

Lemma 8: Given the loss function $Loss = ||H_1 - H_2||_F^2$. Let n, k denote two different nodes in $L(Q)$, and KS_n, KS_k denote their corresponding knowledge segments. Let $h_{e_{k,n}}$ denote the weight of edge between node k and n, and $h_{c_{k,n}}$ denote the weight of edge between KS_k and KS_n. We have

(i) The influence of an edge $A_n(i,j)$ in knowledge segment KS_n can be calculated as
$I(A_n(i,j)) = \sum_{k \in N(n)} -2(h_{e_{k,n}} - h_{c_{k,n}}) \frac{\partial sim(KS_n, KS_k)}{\partial A_n(i,j)}$.

(ii) The influence of a node i in knowledge segment KS_n can be calculated as
$I(N_n(i)) = \sum_{k \in N(n)} -2(h_{e_{k,n}} - h_{c_{k,n}}) \frac{\partial sim(KS_n, KS_k)}{\partial N_n(i)}$.

(iii) The influence of a node attribute j in knowledge segment KS_n can be calculated as
$I(N_n^j(i,i)) = \sum_{k \in N(n)} -2(h_{e_{k,n}} - h_{c_{k,n}}) \frac{\partial sim(KS_n, KS_k)}{\partial N_n^j(i,i)}$.

Proof: We rewrite the loss function as

$$Loss = ||H_1 - H_2||_F^2 = \sum_{i,j} (h_{e_{i,j}} - h_{c_{i,j}})^2$$

Take the derivative, together with Lemma 1, we have

$$\begin{aligned} I(A_n(i,j)) &= \sum_{k \in N(n)} -2(h_{e_{k,n}} - h_{c_{k,n}}) \frac{\partial sim(KS_n, KS_k)}{\partial A_n(i,j)} \\ (N_n(i)) &= \sum_{k \in N(n)} -2(h_{e_{k,n}} - h_{c_{k,n}}) \frac{\partial sim(KS_n, KS_k)}{\partial N_n(i)} \\ I(N_n^l(i,i)) &= \sum_{k \in N(n)} -2(h_{e_{k,n}} - h_{c_{k,n}}) \frac{\partial sim(KS_n, KS_k)}{\partial N_n^l(i,i)} \end{aligned} \tag{14}$$

which completes the proof.

Second, after we find all the key elements, we check the consistency of the semantic matching subgraph according to these key elements. The steps are as follows. For each pair of knowledge segments of the semantic matching subgraph, if their key elements overlapping rate is greater than a threshold (60%), we check the consistency of this pair. Suppose the corresponding triples are $< \mathtt{s}_1, \mathtt{p}_1, \mathtt{o}_1 >$ and $< \mathtt{s}_2, \mathtt{p}_2, \mathtt{o}_2 >$, respectively. We check if $< \mathtt{s}_1, \texttt{isTypeOf}, \mathtt{s}_2 >$ or $< \mathtt{s}_2, \texttt{isTypeOf}, \mathtt{s}_1 >$ is true. If both of them are false, we skip this pair of clues because it does not belong to C3 or C4. Otherwise, we check if $< \mathtt{o}_1, \texttt{isTypeOf}, \mathtt{o}_2 >$ or $< \mathtt{o}_2, \texttt{isTypeOf}, \mathtt{o}_1 >$ is true. If both of them are false, we say this query graph has collective inconsistency. When checking the truthfulness of triples (e.g., $< \mathtt{s}_1, \texttt{isTypeOf}, \mathtt{s}_2 >$, $< \mathtt{s}_2, \texttt{isTypeOf}, \mathtt{s}_1 >$, $< \mathtt{o}_1, \texttt{isTypeOf}, \mathtt{o}_2 >$ and $< \mathtt{o}_2, \texttt{isTypeOf}, \mathtt{o}_1 >$), we use the same method (i.e., transferred information amount in Eq. (13)) as in pairwise comparative reasoning.

5 Experimental Results

In this section, we present the experimental evaluations. All the experiments are designed to answer the following two questions:

- **Q1. Effectiveness.** How effective are the proposed reasoning methods, including both pairwise and collective comparative reasoning methods?
- **Q2. Efficiency.** How fast are the proposed methods?

Two data graphs are used in the experiments: Yago [30] [1] and Covid-19 [2]. Yago [30] is a widely used knowledge graph which contains 12,430,705 triples, 4,295,825 entities and 39 predicates. The Covid-19 data graph contains three types of entities which are `Gene`, `Disease` and `Chemical`. In our experiments, we use a subset of the Covid-19 dataset which contains 55,434 core entities and 5,527,628 triples. We compare our method with 4 baselines, including:

- TransE [1] embeds both the entities and relations in the knowledge graph to a high dimension embedding space, and checks the consistency of a triple according to the embedding distance.
- Jaccard coefficient [14] is a link prediction algorithm which measures the truthfulness of the triple according to the number of common neighbor nodes of the head entity and tail entity.
- Knowledge Linker [2], short for KL, extracts a path between the head entity and tail entity to decide whether the input triple is correct.
- KGMiner [26] extracts a subgraph between the head entity and tail entity to predict the truthfulness of the input clue.

All the experiments are conducted on a moderate desktop with an Intel Core-i7 3.00GHz CPU and 64GB memory. The source code could be found at `https://github.com/lihuiliullh/KompaRe`. For TransE [1] in the experiments, we set the embedding dimension to 64 and use a margin of one and a learning rate of 0.01 for 1,200 epochs.

5.1 Predicate-Predicate Similarity Efficacy

We evaluate the proposed predicate-predicate similarity. Figure 4 presents two examples on Yago dataset. It shows the top-10 most similar predicates with `exports` and `livesIn`, respectively. The font size in Figure 4

[1] It is publicly available at `https://www.mpi-inf.mpg.de/de/departments/databases-and-information-systems/research/yago-naga/yago/downloads`. We use the core version.

[2] The dataset can be found at `http://blender.cs.illinois.edu/covid19/`.

(a) exports (b) livesIn

Figure 4: Top-10 most similar predicates in Yago.

is proportional to the predicate-predicate similarity value. The top similar predicates w.r.t. `exports` by our method include `imports`, `hasOfficialLanguage`, `dealsWith`, all of which have a high similarity with `exports`. They all provide specific semantic information about `exports`. Likewise, the top similar predicates w.r.t. `livesIn` include `wasBornIn`, `isCitizenOf`, `diedIn`, all of which are closely related to `livesIn`. These results showcase that the proposed TF-IDF based method can effectively measure the similarity between different predicates.

Table 5 shows the predicate similarity between `isTypeOf` and other predicates.

Table 5: Predicate similarity of `isTypeOf` with others

predicate	sim	predicate	sim	predicate	sim	predicate	sim
isCitizenOf	0.840	isLeaderOf	0.955	isAffiliatedTo	0.808	isPoliticianOf	0.917
livesIn	0.972	owns	0.945	exports	0.706	dealsWith	0.697
hasCapital	0.786	command	0.216	happenedIn	0.767	participatedIn	0.869
worksAt	0.752	isLocatedIn	0.870				

5.2 Pair-wise Comparative Reasoning

Here, we evaluate the effectiveness of the proposed pair-wise comparative reasoning. Ten query sets are used in the experiments. For each positive query set, it contains a set of queries which describe the true claim, while for each negative query set, it contains a set of queries which describe the false claim. For example, in query set "Birth Place", <`Alan Turing`, `wasBornIn`, `Maida Vale`> and <`Alan Turing`, `wasBornIn`, `United Kingdom`> is a positive query pair, while <`Alan Turing`, `wasBornIn`, `Maida Vale`> and <`Alan Turing`, `wasBornIn`, `Canada`> is an negative query pair. The positive queries are generated by sampling some true claims in the knowledge graph. The negative queries are generated by substituting one subject

Table 6: Accuracy of pair-wise comparative reasoning.

Dataset	# of queries	TransE	Jaccard	KL	KGMiner	Neural Network Based	Graph Kernel Based
Family members positive	300	0.682	0.831	0.618	0.983	**1.000**	0.944
Family members negative	300	0.335	0.169	**1.000**	**1.000**	0.000	0.941
Graduated college positive	300	0.686	0.335	0.502	0.769	**0.879**	0.794
Graduated college negative	300	0.626	0.993	0.947	0.901	0.367	**0.994**
Live place positive	300	0.567	0.415	0.489	**0.834**	0.086	0.762
Live place negative	300	0.802	0.585	**0.907**	0.900	0.732	0.888
Birth place positive	300	0.590	0.435	0.537	0.698	0.656	**0.800**
Birth place negative	300	0.845	**1.000**	0.973	0.927	0.719	0.927
Work place positive	300	**0.751**	0.319	0.445	0.698	0.700	0.720
Work place negative	300	0.624	0.994	0.942	0.927	0.803	**0.995**
$mean \pm std$	-	0.651 ± 0.424	0.608 ± 0.302	0.736 ± 0.221	0.864 ± 0.105	0.594 ± 0.333	$\mathbf{0.877 \pm 0.095}$

of the positive queries. The accuracy is defined as $\frac{N}{M}$ where N is the number of queries correctly classified by pair-wise comparative reasoning and M is the total number of queries. When checking the consistency of a query pair $< \mathtt{s}_1, \mathtt{p}_1, \mathtt{o}_1 >$ and $< \mathtt{s}_2, \mathtt{p}_2, \mathtt{o}_2 >$, because none of the baseline methods is designed for pair-wise comparative reasoning, we use them to check each triple in the pair, if any triple is classified as false, this query pair is treated as false. Otherwise, we further check the truthness of $< \mathtt{o}_1, \mathtt{isTypeOf}, \mathtt{o}_2 >$ and $< \mathtt{o}_2, \mathtt{isTypeOf}, \mathtt{o}_1 >$, if one of them is classified as consistency, this query pair is treated as consistency. Table 6 gives the detailed results.

As we can see, Graph Kernel based method and KGMiner [26] have the highest accuracy most of the time. But Graph Kernel based method has the highest average accuracy and the lowest variance compared with other methods.

5.3 Neural Network Based Pair-wise Comparative Reasoning

We conduct more experiments in this section to test the effectiveness of Neural Network Based Pairwise comparative reasoning. Six binary classifiers are used in the experiment. Table 7 shows the details of each classifier and their performance on different query sets. As we can see, different methods have different performances. Logistic Regression has the highest average accuracy and K-Nearest Neighbor has the highest average recall.

Table 7: Accuracy, recall and precision of neural network based pair-wise comparative reasoning

Model \ Dataset	Family members			Graduated college			Live place			Birth place			Work place			$mean \pm std$		
measurements	Acc	Prec	Recall	Acc	Prec	Recall	Acc	Prec	Recall	Acc	Prec	Recall	Acc	Prec	Recall	Acc	Prec	Recall
# of queries	506			499			560			470			540			-		
Logistic Regression	0.480	0.519	0.491	0.693	0.660	0.729	0.619	0.732	0.594	0.674	0.816	0.645	0.743	0.786	0.733	**0.642±0.090**	0.703±0.106	0.639±0.091
SVM	0.431	0.154	0.364	0.416	0.113	0.333	0.611	0.768	0.581	0.516	1.000	0.516	0.688	0.911	0.637	0.532±0.104	0.589±0.380	0.486±0.119
Decision Trees	0.422	0.519	0.443	0.713	0.623	0.786	0.619	0.696	0.600	0.611	0.694	0.607	0.716	0.750	0.712	0.616±0.107	0.656±**0.080**	0.629±0.116
Random Forest	0.314	0.462	0.364	0.624	0.528	0.683	0.673	0.714	0.656	0.684	0.673	0.702	0.706	0.661	0.740	0.600±0.146	0.608±0.096	0.629±0.135
Naive Bayes	0.461	0.615	0.478	0.624	0.830	0.603	0.522	0.839	0.511	0.632	0.878	0.597	0.725	0.911	0.671	0.593±0.092	**0.815**±0.104	0.572±**0.069**
K-Nearest Neighbor	0.392	0.500	0.419	0.683	0.774	0.672	0.646	0.589	0.660	0.695	0.673	0.717	0.761	0.821	0.742	0.636±0.127	0.672±0.118	**0.642**±0.115

Table 8 shows performance of the neural network based method on each positive and negative query dataset. Based on the results in the table, we can conclude that the accuracy of the neural network based model is lower than that of other models, e.g., KGMiner and Graph Kernel Based method.

Table 8: Accuracy of neural network based pair-wise comparative reasoning

Dataset \ Model	# of queries	Logistic Regression	Support Vector Machines	Decision Trees	Random Forest	Naive Bayes	K-Nearest Neighbor
Family members positive	258	0.519	0.577	0.481	0.481	0.615	0.500
Family members negative	248	0.440	0.420	0.340	0.260	0.300	0.280
Graduated college positive	261	0.660	0.868	0.623	0.547	**0.830**	0.774
Graduated college negative	238	0.729	0.208	**0.812**	**0.708**	0.396	0.583
Live place positive	277	0.732	**1.000**	0.643	0.679	0.839	0.589
Live place negative	283	0.509	0.632	0.526	0.632	0.211	0.702
Birth place positive	244	**0.816**	**1.000**	0.653	0.653	0.878	0.673
Birth place negative	226	0.522	0.000	0.522	0.674	0.370	0.717
Work place positive	277	0.786	0.946	0.696	0.679	0.911	**0.821**
Work place negative	263	0.698	**1.000**	0.660	0.755	0.528	0.698
$mean \pm std$	-	0.641 ± 0.126	**0.665** ± 0.343	0.596 ± **0.125**	0.607 ± 0.138	0.588 ± 0.250	0.634 ± 0.148

5.4 Collective Comparative Reasoning

We test collective comparative reasoning method on Yago dataset, using 6 query sets . Different from the queries of pair-wise comparative reasoning which only contain two edges, each query of collective comparative reasoning contains 3 edges. For example, in query set "live Place", <`Barack Obama`, `livesIn`,

Table 9: Accuracy of collective comparative reasoning.

Dataset	# of queries	TransE	Jaccard	KL	KGMiner	Kompare
Birth place positive	300	0.542	0.418	0.389	0.678	**0.795**
Birth place negative	300	0.465	**0.996**	0.968	0.970	0.829
Live place positive	300	0.448	0.451	0.465	0.635	**0.989**
Live place negative	300	0.558	**1.000**	0.860	0.924	0.743
Graduated college positive	300	0.488	0.269	0.335	0.585	**0.963**
Graduated college negative	300	0.545	**0.996**	0.928	0.907	0.829
$mean \pm std$	-	0.508 $\pm$ **0.045**	0.688 $\pm$ 0.313	0.658 $\pm$ 0.265	0.783 $\pm$ 0.155	**0.858** $\pm$ 0.089

Washington,D.C.>, <Barack Obama, is, United States Senate Barack Obama> and <United States Senate Barack Obama, livesIn, United States> is a positive query triad, while <Barack Obama, livesIn, Washington,D.C.>, <Barack Obama, is, United States Senate Barack Obama> and <United States Senate Barack Obama, livesIn, Canada> is an negative query triad. The definition of the accuracy is the same as the previous section. Following the setting of pair-wise reasoning, when checking the consistency of the query graph $< \mathtt{s}_1, \mathtt{p}_1, \mathtt{o}_1 >$, $< \mathtt{s}_1, \mathtt{is}, \mathtt{s}_2 >$ and $< \mathtt{s}_2, \mathtt{p}_2, \mathtt{o}_2 >$, we use baseline methods to check the truthness of this query triad, if any edge is classified as false, this query triad is treated as false. Otherwise, we further check the truthness of $< \mathtt{o}_1, \mathtt{isTypeOf}, \mathtt{o}_2 >$ and $< \mathtt{o}_2, \mathtt{isTypeOf}, \mathtt{o}_1 >$, if one of them is classified as true , this query pair is treated as consistency. Table 9 gives the detailed results. As we can see, Jaccard [14] prefers to classify all queries as inconsistency and has the largest variance. TransE [1] has the lowest variance, but its average accuracy is very low. KOMPARE has the highest accuracy most of the time. It also has the highest average accuracy, and the second lowest variance.

Table 10: Accuracy of collective comparative reasoning for Covid-19.

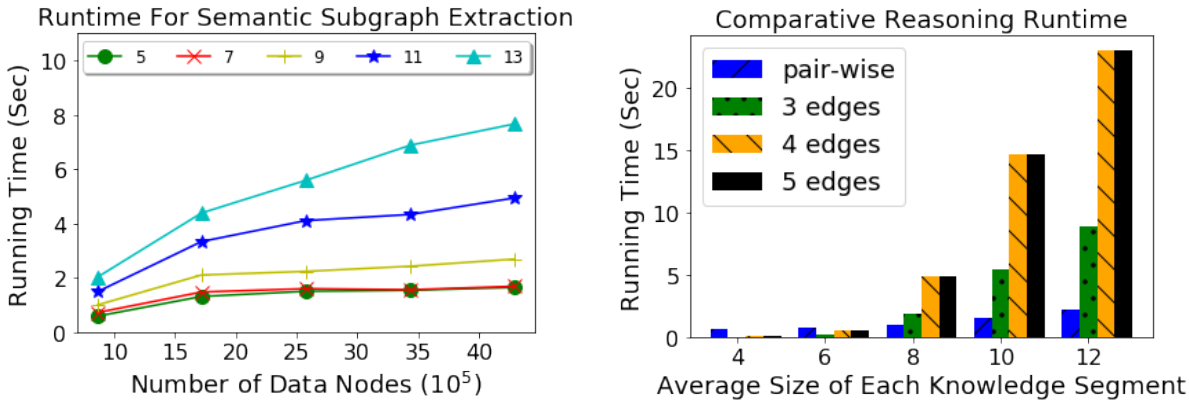
Dataset	# of queries	TransE	Jaccard	KL	KGMiner	Kompare
Positive	36	0.667	0.611	**1.000**	0.694	**1.000**
Negative	36	0.528	0.361	0.722	0.553	**0.863**
Average accuracy	-	0.598 $\pm$ 0.071	0.486 $\pm$ 0.126	0.861 $\pm$ 0.138	0.623 $\pm$ 0.071	**0.932** $\pm$ **0.063**

We further provide experimental results on Covid-19 dataset. We use queries which contain connections between drugs and genes/chemicals related to covid-19.[3] Among all these queries, we use queries which contain less than 8 nodes, and treat them as positive queries. For each of the positive queries, we randomly select one node inside the query and substitute it with a randomly selected entity in the data graph, and treat the new query as the negative query. For all the baseline methods, we use them to check all the edges inside the query, if any edge is classified as false, the whole query is treated as false. Table 10 shows the accuracy of different methods. As we can see, KOMPARE has the highest accuracy on both the positive and negative datasets, it also has the highest average accuracy and the lowest variance compared with other baseline methods.

5.5 Efficiency Results

The runtime of knowledge segment extraction depends on the size of the underlying knowledge graphs. Among the two types of knowledge segments (edge-specific knowledge segment and subgraph-specific knowledge segment), subgraph-specific knowledge segment is most time-consuming. Figure 5(a) shows that its runtime scales sub-linearly w.r.t. the number of nodes in the knowledge graph. Different lines show the runtime w.r.t. different query graph size. Figure 5(b) shows the runtime of comparative reasoning, where 'Pair-wise' refers to the pairwise comparative reasoning, and the remaining bars are for collective comparative reasoning with 3, 4 and 5 edges in the query graphs respectively. Note that the runtime of comparative reasoning only depends on the

[3]The query graphs can be found at http://blender.cs.illinois.edu/covid19/visualization.html.



(a) Subgraph-specific KS extraction

(b) Comparative reasoning

Figure 5: Runtime of KOMPARE

size of the the corresponding knowledge segments which typically have a few or a few tens of nodes. In other words, the runtime of comparative reasoning is independent of the knowledge graph size.

6 Related Work

A - Knowledge Graph Search. Many efforts have been made for searching and browsing large knowledge graphs. Wang et al. [36] proposed a Bayesian probability model combined with random walks to find the most similar concepts for a given query entity. Wu et al. [29] discovered that the background knowledge graph can be described by many small-sized patterns. They developed an effective mining algorithm to summarize the large knowledge graph according to small-sized patterns. Yang et al. [39] found that due to the lack of insight about the background knowledge graph, it is often hard for a user to precisely formulate a query. They developed a user-friendly knowledge graph search engine to support query formation and transformation. Jayaram et al. [10] proposed a knowledge graph query system called GQBE. Different from other graph query systems, GQBE focuses on entity tuple query which consists of a list of entity tuples. Zhang et al. [40] developed a comprehensive multi-modality knowledge extraction and hypothesis generation system which supports three types of queries, including (1) class-based queries (2) zero-hop queries and (3) graph-queries.

B - Fact Checking on Knowledge Graph. In 2015, GL Ciampaglia et al. [3] show that the complexities of human fact checking can be approximated quite well by finding the shortest path between concept nodes under properly defined semantic proximity metrics on knowledge graphs. The authors evaluate tens of thousands of claims on knowledge graphs extracted from Wikipedia. Many research works follow this direction with different techniques. Baoxu et al. [27] model the fact checking problem as a link-prediction task in a knowledge graph, and present a discriminative path-based method for fact checking in knowledge graphs. Shiralkar et al. [28] adopts k-shortest paths approach to construct a knowledge stream (KS) between two entities as the background knowledge for fact checking. Lin et al. [15] introduce ontological patterns in fact checking for semantic and topological constraints. These constraints are represented as subgraph patterns which are used for query in the knowledge graph. In order to obtain the ground truth of contextualized claim, Tchechmedjiev et al. release a large, up-to-date and queryable corpus of structured information about claims and related metadata for fact checking research, named ClaimsKG [32].

C - Knowledge Graph Reasoning. Generally speaking, there are two types of knowledge graph reasoning methods, including (1) embedding based approaches and (2) multi-hop approaches. For the former, the main idea is to learn a low dimensional vector for each entity and predicate in the embedding space, and use these embedding vectors as the input of the reasoning tasks (e.g., [1], [31], [12], [35]). For the latter, the main

idea is to learn missing rules from a set of relational paths sampled from the knowledge graph (e.g., [13], [37], [22]). Many effective reasoning methods have been developed for predicting the missing relation (i.e., link prediction) or the missing entity (i.e., entity prediction). In link prediction, given the 'subject' and the 'object' of a triple, it predicts the existence and/or the type of relation. For example, TransE [1] learns the low dimensional embedding of both entities and predicates in the knowledge graph; TransR [16] learns the embedding of entities and predicates in two separate spaces. The learned embedding (either by TransE or TransR) can be used for both link predication and entity predication. In entity prediction, given the 'subject' and the 'predicate' of a triple, it predicts the missing 'object'. For example, GQEs [12] embeds the graph nodes in a low dimensional space, and treats the logical operators as learned geometric operations.

In recent years, knowledge graph reasoning has demonstrated strong potential for computational fact checking. Given a claim in the form of a triple of the knowledge graph, it reasons whether the claim is authentic or falsified. For example, in [24], the authors focused on checking the truthfulness of a given triple/claim, by first transforming the knowledge graph into a weighted directed graph, and then extracting a so-called knowledge stream based on maximum flow algorithm. It is worth mentioning that the extracted knowledge stream can be viewed as an edge-specific knowledge segment in KOMPARE. In [25], an alternative method was developed to detect fake claims by learning the discriminative paths of specific predicates. Different from [24], this is a supervised reasoning method since it requires different training datasets for different predicates. If the predicate in the claim does not exist in the training data, which is likely to be the case for detecting falsified claims in emerging news, the algorithm becomes inapplicable. As mentioned before, these methods belong to point-wise reasoning. Therefore, they might fall short in detecting the semantic inconsistency between multiple claims which can be solved by knowledge graph comparative reasoning.

7 Conclusions

In this paper, we present the problem definition and algorithms for knowledge graph comparative reasoning. Comparative reasoning aims to complement and expand the existing point-wise reasoning over knowledge graphs by inferring commonalities and inconsistencies of multiple pieces of clues. We propose several methods to tackle comparative reasoning. At the heart of the proposed methods are a suite of core algorithms, including predicate-predicate similarity and semantic subgraph matching for knowledge segment extraction; neural network and influence function, commonality rate, transferred information amount for both pairwise reasoning and collective reasoning. The experimental results demonstrate that the proposed methods (1) can effectively detect semantic inconsistency, and (2) scales near linearly with respect to the knowledge graph size.

References

[1] B. Antoine, U. Nicolas, G. Alberto, W Jason, and Y. Oksana. Translating embeddings for modeling multi-relational data. In NIPS '13, NIPS '13, pages 2787–2795.

[2] Giovanni Luca Ciampaglia, Prashant Shiralkar, and Rocha. Computational fact checking from knowledge networks. 2015.

[3] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. Computational fact checking from knowledge networks. PloS one, 10(6):e0128193, 2015.

[4] Limeng Cui, Suhang Wang, and Dongwon Lee. Same : Sentiment-aware multi-modal embedding for detecting fake news. 2019.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[6] Boxin Du, Lihui Liu, and Hanghang Tong. Sylvester tensor equation for multi-way association. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '21, page 311–321, New York, NY, USA, 2021. Association for Computing Machinery.

[7] C. Faloutsos, K. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In KDD '04, pages 118–127, New York, NY, USA, 2004. ACM.

[8] S. Freitas, N. Cao, Y. Xia, D. H. P. Chau, and H. Tong. Local partition in rich graphs. BigData '19, pages 1001–1008, Dec 2018.

[9] C. Giovanni, S. Prashant, R. Luis, B. Johan, M. Filippo, and F. Alessandro. Computational fact checking from knowledge networks. PloS one, 10, 01 2015.

[10] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri. Querying knowledge graphs by example entity tuples. 27(10):2797–2811, Oct 2015.

[11] Y. Koren, S. North, and C. Volinsky. Measuring and extracting proximity in networks. KDD '06, pages 245–255, New York, NY, USA, 2006. ACM.

[12] H. William L., B. Payal, Z. Marinka, J. Dan, and L. Jure. Embedding logical queries on knowledge graphs. NIPS'18, page 2030–2041, Red Hook, NY, USA, 2018.

[13] Ni L, Tom M, and William W. C. Random walk inference and learning in a large scale knowledge base. EMNLP '11, USA, 2011.

[14] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. CIKM '03.

[15] Peng Lin, Qi Song, and Yinghui Wu. Fact checking in knowledge graphs with ontological subgraph patterns. Data Science and Engineering, 3(4):341–358, 2018.

[16] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. AAAI'15. AAAI Press, 2015.

[17] L. Liu, B. Du, and H. Tong. Gfinder: Approximate attributed subgraph matching. BigData '19, Dec 2019.

[18] Lihui Liu, Boxin Du, Yi Ren Fung, Heng Ji, Jiejun Xu, and Hanghang Tong. Kompare: A knowledge graph comparative reasoning system. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '21, page 3308–3318, New York, NY, USA, 2021. Association for Computing Machinery.

[19] Lihui Liu, Boxin Du, Heng Ji, and Hanghang Tong. A knowledge graph reasoning prototype. NeurIPS (demo track), 2020.

[20] Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. Joint knowledge graph completion and question answering. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, page 1098–1108, New York, NY, USA, 2022. Association for Computing Machinery.

[21] Kai Nakamura, Sharon Levy, and William Yang Wang. r/fakeddit: A new multimodal benchmark dataset for fine-grained fake news detection. CoRR, abs/1911.03854, 2019.

[22] D. Rajarshi, N. Arvind, B. David, and M. Andrew. Chains of reasoning over entities, relations, and text using recurrent neural networks. ACL '17, April 2017.

[23] Shane Roach, Connie Ni, Alexei Kopylov, Tsai-Ching Lu, Jiejun Xu, Si Zhang, Boxin Du, Dawei Zhou, Jun Wu, Lihui Liu, Yuchen Yan, Jingrui He, and Hanghang Tong. Canon: Complex analytics of network of networks for modeling adversarial activities. In 2020 IEEE International Conference on Big Data (Big Data), pages 1634–1643, 2020.

[24] Prashant S, Alessandro F, Filippo M, and Giovanni C. Finding streams in knowledge graphs to support fact checking. pages 859–864, 11 2017.

[25] B. Shi and T. Weninger. Discriminative predicate path mining for fact checking in knowledge graphs. Know.-Based Syst., 104(C):123–133, July 2016.

[26] Baoxu Shi and Tim Weninger. Discriminative predicate path mining for fact checking in knowledge graphs.

[27] Baoxu Shi and Tim Weninger. Discriminative predicate path mining for fact checking in knowledge graphs. Knowledge-based systems, 104:123–133, 2016.

[28] Prashant Shiralkar, Alessandro Flammini, Filippo Menczer, and Giovanni Luca Ciampaglia. Finding streams in knowledge graphs to support fact checking. In 2017 IEEE International Conference on Data Mining (ICDM), pages 859–864. IEEE, 2017.

[29] Q. Song, Y. Wu, P. Lin, L. X. Dong, and H. Sun. IEEE Transactions on Knowledge and Data Engineering, 30(10):1887–1900, Oct 2018.

[30] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. WWW '07. Association for Computing Machinery, 2007.

[31] Z. Sun, Z. Deng, J. Nie, and J. Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. ArXiv, abs/1902.10197, 2019.

[32] Andon Tchechmedjiev, Pavlos Fafalios, Katarina Boland, Malo Gasquet, Matthäus Zloch, Benjamin Zapilko, Stefan Dietze, and Konstantin Todorov. Claimskg: a knowledge graph of fact-checked claims. In International Semantic Web Conference, pages 309–324. Springer, 2019.

[33] Hanghang Tong and Christos Faloutsos. Center-piece subgraphs: Problem definition and fast solutions. KDD '06, pages 404–413, New York, NY, USA, 2006. ACM.

[34] Hanghang Tong, Christos Faloutsos, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. ICDM '06, pages 613–622, Washington, DC, USA, 2006. IEEE Computer Society.

[35] William Yang Wang and William W. Cohen. Learning first-order logic embeddings via matrix factorization. IJCAI'16, page 2132–2138. AAAI Press, 2016.

[36] Z. Wang, K. Zhao, H. Wang, X. Meng, and J. Wen. Query understanding through knowledge-based conceptualization. IJCAI'15, pages 3264–3270. AAAI Press, 2015.

[37] W Xiong, T Hoang, and W Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In EMNLP, 2017.

[38] Yuchen Yan, Si Zhang, and Hanghang Tong. Bright: A bridging algorithm for network alignment. In Proceedings of the Web Conference 2021, WWW '21, page 3907–3917, New York, NY, USA, 2021. Association for Computing Machinery.

[39] Shengqi Yang, Yinghui Wu, Huan Sun, and Xifeng Yan. Schemaless and structureless graph querying. Proc. VLDB Endow., 7(7):565–576, March 2014.

[40] T. Zhang, G. Shi, L. Huang, and D. Lu and. GAIA - A multi-media multi-lingual knowledge extraction and hypothesis generation system. TAC' 18, 2018.

[41] Q. Zhou, L. Li, N. Cao, L. Ying, and H. Tong. adversarial attacks on multi-network mining: problem definition and fast solutions. ICDM '19, Dec 2019.

Distilling Causal Metaknowledge from Knowledge Graphs

Yuan Meng, Yancheng Dong, Shixuan Liu, Chaohao Yuan, Yue He, Jian Pei, Peng Cui

Abstract

In recent years, the explosive increase of information facilitates the massive knowledge graphs, which in turn increase burden of people to understand and leverage the regularity behind these superficial facts. Therefore, the metaknowledge, defined as the knowledge about knowledge, is proposed to identify complex processes of knowledge production and consumption. Unfortunately, even though the current correlation-based rule mining methods in knowledge graph distill the rule-formed metaknowledge, they can not explain the processes of knowledge production. In this paper, we focus on capturing the metaknowledge with causality which is generally regarded as one of the most promising techniques to reveal the interactions between components in the complex system. To the best of our knowledge, this is the first attempt to interpret the knowledge graph from the causal perspective.

For this purpose, we propose a causal metaknowledge method for link prediction, which achieves entity-level link prediction by discovering concept-level topological causality. Specifically, we first formalize causal metaknowledge as causal rule, following the form of logical rule. Then, we transform the relational data into propositional data to learn the causal relationships between topological structures. And an efficient algorithm for discovering local causal relationships is proposed using the d-seperation criterion. Eventually, the causal rules generated based on the mined relationships are used for link prediction. Both simulation-based and real data-based experiments demonstrate the effectiveness of the proposed approach, especially under the Out-of-Distribution(OoD) settings.

1 Introduction

In the era of information explosion, knowledge graph (KG) is a powerful representation for integrating billions of available relational facts, based on observational low-level knowledge in the world, to encapsulate the rich relationships of entities [17, 45]. Although the massive knowledge can benefit various downstream applications, *e.g.* query answering [42, 23, 4], recommendation systems [41, 40, 22], yet to better understand, exploit, and complete these underlying knowledge, it is necessary to explore the intrinsic principle of the emergence of this factual knowledge. For this purpose, the concept of meta-knowledge is proposed and defined as the *knowledge about knowledge* [6].

Current rule mining methods in the KG literature attempt to mine meta-knowledge, in the form of association rules, via correlation analysis represented by frequency analysis [9, 8, 27]. These association rules can be used for downstream tasks such as knowledge graph completion, and question answering. However, association does not imply causation [1]. Fortunato et al. points out that causality is necessary to identify the fundamental drivers

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

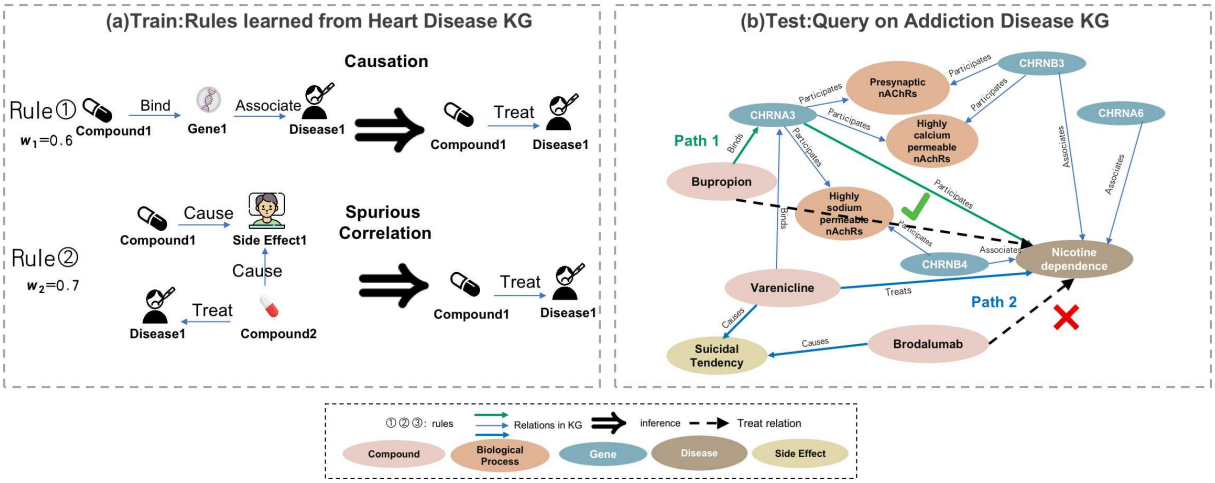


Figure 1: Motivation illustration. Consider a scenario to learn rules for inferring the `Treat` relation between compounds and diseases on a heart disease KG (top), and thereby discover novel drugs for treating nicotine addiction on an addiction disease KG (bottom). On the heart disease KG, drugs that treat the same disease often share the same side effects. The correlation-based approach establishes a strong spurious correlation between the shared side-effect information and the `treat` relation. In contrast, the underlying cause of the `treat` shows a weak association (Rule ①). Therefore, when the above rule is migrated to addiction disease KG (drugs that treat the same basic kind of disease often do not have shared side effects), false prediction could be resulted (e.g., Brodalumab is more likely to be prescribed for nicotine addiction, instead of Bupropion).

of knowledge [7]. The correlation-based method may lead to spurious correlations between the body and head of rule, which can not be generalized to new environments.

Here we take the KG-based drug repurposing task as an example (shown in Fig. 1). Given the heart disease KG as training data, traditional rule mining methods may rely on two rules ① and ② to predict the `Treat` relation. As heart disease drugs entail similar side effects, the confidence (weight) of Rule ②, calculated based on correlation, is greater than that of Rule ① (0.7 versus 0.6). However, the localization of the drug to the target protein produced by the disease gene, as indicated in Rule ①, is the recognized mechanism for physicians to prescribe drugs for the disease [15]. This phenomenon, typical of spurious correlations, is due to the scarcity of genetic information and the abundance of side-effect facts accompanying the data collection process. Therefore, these weighted rules could produce false KG completion results as the environment shifts. Fig. 1(b) visualizes such testing process where the learned rules from heart disease KG are used to answer queries from addiction disease KG. Both nicotine withdrawal drug (Varenicline) and psoriasis drug (Brodalumab) are known to cause the side effect suicidal tendencies. However, the available drug for nicotine withdrawal Bupropion (which binds the gene that participates in nicotine) is not. With the mined rules in Fig. 1(a), physicians could falsely prescribe Brodalumab (Path 2) as a new treatment for nicotine withdrawal, instead of Bupropion (Path 1). If we can learn stable relationships (such as causality) between predicted features and predicted targets, such effect of spurious correlations can be eliminated.

In this paper, we propose a method that learns rules from the causal perspective to ensure strong generalization ability whilst retaining decent interpretability. Specifically, we are concerned with understanding how KG links are generated, through causal discovery. There are two major challenges in this problem: 1) efficiency and 2) proper metrics. For the former, the complex topological structures between massive entity pairs could induce thousand-scale rule space with barely ten relations, posing challenges for both score-based and constraint-based causal discovery approaches. The complexity of the constraint-based technique increases exponentially with the number of nodes, whereas the score-based approach creates an NP-hard problem [19]. For the latter, rule-mining

algorithms generally require specific metrics, such as support rate, as the weights for inference. Therefore, we also need to design a metric to measure the strength of each causal relationship.

In this work, we first formulate causal meta-knowledge with the concept of *causal rule*, on which we further introduce several constraints to reduce the search space. Then, we propose the CMLP (Causal Metaknowledge-based Link Prediction) algorithm, which integrates efficient causal rule discovery approach and causation-based link prediction method. Specifically, we first introduce the concept of rule-induced variable, which uses relation paths to describe the topological structure of entities, and map the graphs into quantified samples with the designed assignment function. Further, we observe that the whole causal structure is not necessary for specific link prediction task, but only the part of the structure related to the predicted relation. Therefore, we design an efficient method based on d-seperation to achieve local causal discovery. Meanwhile, the causal strength based on conditional dependence can also be generated as the weights of learned causal rules. Finally, the predictions can be ranked from weighted causal rules.

Contributions. Our main contributions can be summarized as follows: *(i)* This is the first work that aims at improving link prediction in KG by causal inference to eliminate the effect of spurious correlation, as evidenced in traditional methods. *(ii)* This work introduces CMLP that learns a link predictor based on discovering causal meta-knowledge. *(iii)* CMLP outperforms other competitive baselines on link prediction tasks under Out-of-Distribution (OoD) setting. Furthermore, we analyze the learned meta-knowledge for insights on the mechanism of the applications.

2 Preliminaries and Problem Statement

2.1 Definitions and Notations

In this paper, we follow the definition of knowledge graph as in [17]:

Definition 2.1: A **Knowledge Graph (KG)** is defined as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$, where $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{F}$ are sets of entities, relations and facts, respectively. Every fact is a triple $(e_h, R, e_t) \in \mathcal{F}$, where $e_h, e_t \in \mathcal{E}$ and $R \in \mathcal{R}$ are head entity, tail entity and the relation between entities, respectively. Without loss of generality, we simultaneously represent a fact as $R(e_h, e_t)$.

First-order logic (FOL) offers a pivotal way to represent real-world knowledge for reasoning. Horn rules, as a special and typical case of FOL rules, propose to represent a target relation by a body of conjunctive relations.

Definition 2.2: A **Horn Rule**, generally chain-like, is given as,

$$R_h(x, y) \leftarrow R_{b_1}(x, z_1) \circ \cdots \circ R_{b_l}(z_{l-1}, y)$$

where, $R_h(x, y)$ signifies the rule head (target relation) that we wishes to reason and $R_{b_1}(x, z_1) \circ \cdots \circ R_{b_l}(z_{l-1}, y)$ is the rule body (relation path). For simplicity, we denote a Horn rule as $R_h : \mathbf{R_b}$, where $\mathbf{R_b} = [R_{b_1}, \cdots, R_{b_l}]$. To reason R_h, the size of the rule space is $|\mathcal{B}_h|$. Every **closed path** of such Horn rule is required to: 1) connect (x, y) via the rule body, which is a sequence of relations $\mathbf{R_b}$, and 2) ensure (x, y) are accessible directly via the target relation R_h. Closed paths are also known as **rule instances**.

2.2 Problem Statement

The goal of this work is to learn the **causal rule** that is formalized as the horn rule. Specifically, the objective of traditional logical rule learning is to assign a plausibility score $\mathbf{S}(R_h|\mathbf{R_b})$ to each rule in the discovered rule space, which can be subsequently aggregated to answer queries about the KG. Currently, plausibility scores are defined over closed paths (e.g., the PCA confidence for AMIE [10]), which are correlational observations.

We have demonstrated that these scores are prone to spurious correlations and therefore result in inaccurate predictions under OoD settings, in Sec. 1. Therefore the other aim is to give a plausibility score based on causal strength.

In this paper, the causal rules are mined for link prediction in KG. We follow the commonly accepted problem definition of link prediction in KG [32, 39]: given an observed KG $\mathcal{G}$ with missing facts, our goal is to predict the correct entity for an given query $(e_h, R_h, ?)$ (or $(?, R_h, e_t)$).

3 Proposed Method: CMLP

In this section, we introduce the proposed approach CMLP which learns causal rules for KG link prediction. CMLP first transforms the relational data into the propositional data to conduct statistical analysis (Sec. 3.1). Then CMLP presents a local causality identification algorithm based on the d-seperation criterion to efficiently mines interpretable causal rules (Sec. 3.2). Finally, a specific causation-based score is applied in predictor to answer the queries with learned causal rules (Sec. 3.3). The pipeline of CMLP is illustrated in Fig. 2.

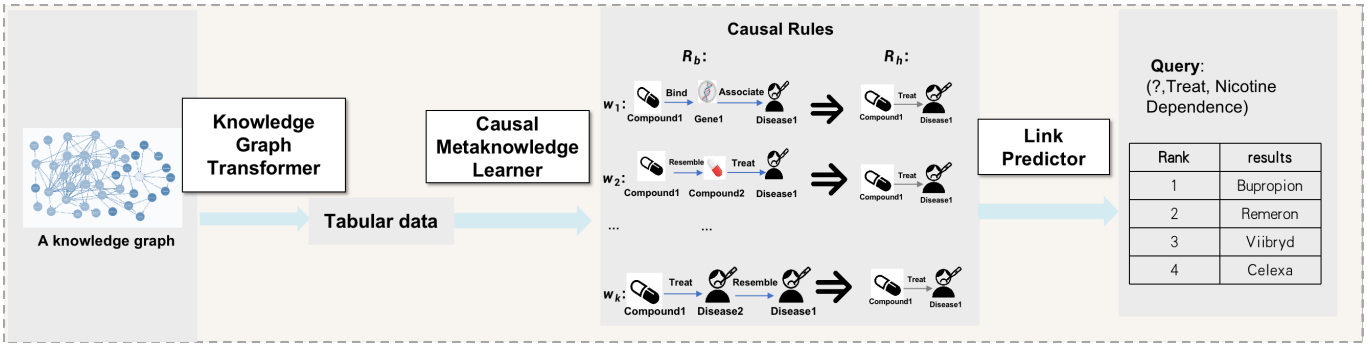


Figure 2: The framework of CMLP. Particularly, CFLP first transforms the relational data into propositional data for better statistical analysis. Then it mines interpretable causal rules, which can be interpreted as a kind of metaknowlege[6]. Finally, a plausibiliy score derived from the causality test is applied in predictor to rank the answers of the given query.

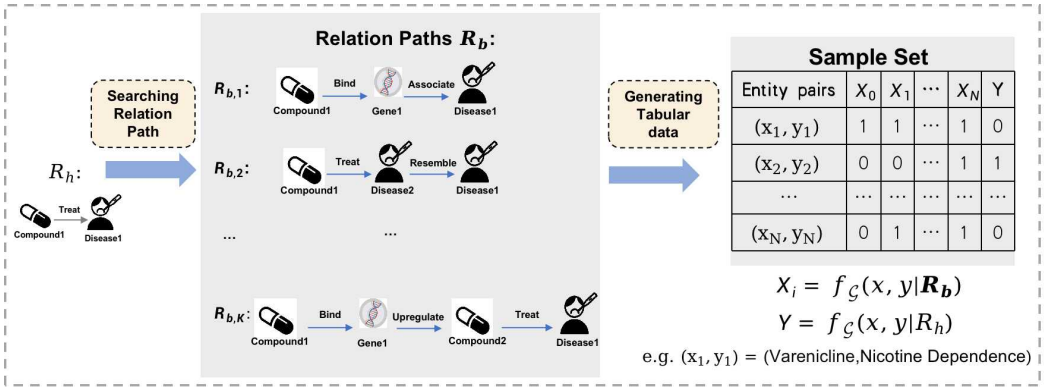


Figure 3: The process of knowledge graph transformer

3.1 Knowledge Graph Transformer

Traditional causal discovery algorithms are defined on propositional data, with well-defined variables and samples, which do not exist in relational data like KGs. Therefore, we give the definition and scope of the variables we study in the causal discovery phase by mapping the potential causes and queried relations into variables. Then we give the practical approach for transforming KG into tabular data, whose horizontal axis are the variables we defined. The process is shown schematically in Fig. 3

3.1.1 Causal variables in KG

The causal rule can be interpreted as a description of causal relationship between the body and the head. Naturally, we formulate variables based on the elements of rules.

Definition 3.1: For entity pair (x, y), its **Rule-induced Variable** $X_k = f_{\mathcal{G}}(x, y|{\mathbf{R_b}}^k)$, where ${\mathbf{R_b}}^k$ corresponds to the rule body in the k-th rule $R_h : {\mathbf{R_b}}^k (k \in \{1, ..., |\mathcal{B}_h|\})$. The assignment function $f_{\mathcal{G}}(\cdot|\mathbf{R_b})$ can be either connectivity feature or path count for $\mathbf{R_b}$ in KG $\mathcal{G}$.

The head of rule also induces a special variable $Y = f_{\mathcal{G}}(x, y|R_h)$. In the real link prediction task, the queries are normally on a specific relation, such as `Treat` in drug repurposing. Therefore, the causal rule mining problem is to discovery the causal relationship between variables $X_k = f_{\mathcal{G}}(x, y|{\mathbf{R_b}}^k), k \in \{1, ..., |\mathcal{B}_h|\}$ and variable $Y = f_{\mathcal{G}}(x, y|R_h)$. Then we introduce the practical approach that we transform the KG into tabular data for causality analysis.

3.1.2 Transforming knowledge graph into propositional data

(1) Step-1: Searching candidate causes X. According to definition 2.2, any rule-induced variable X, which is defined on entity pair(x, y) and seeks to help reasoning over R_h, is a valid candidate cause for $Y = f_{\mathcal{G}}(x, y|R_h)$. So we find all the candidate causes by searching all the paths between entity pairs (x, y), which have the relation R_h between them. There are many well-studied path finding algorithms, which can search the paths under different types of constraints, such as Dijkstra's algorithm [18], A* search [5], best-first search [14], etc. In the experiments, we adopt the best-first search algorithm. Since the number of candidate causes can be the power level of the number of relation types, we require that the length of the path is no more than ℓ, where ℓ is the hyper-parameters. In the experiments of this paper, we set ℓ as 3. (2) Step-2: generating samples. In this paper, we use the connectivity as the assignment function to get quantitative samples.

Definition 3.2: the **binary assignment function of rule-induced variable** is as following:

$$f_{\mathcal{G}}(e_h, e_t|{\mathbf{R_b}}^k) = \mathbb{1}_{\text{con}}(e_h, e_t|{\mathbf{R_b}}^k)$$

where $\mathbb{1}_{\text{con}}(e_h, e_t|{\mathbf{R_b}}^k) \in \{0, 1\}$ checks whether there exists a path instance of $\mathbf{R_b^k}$ between e_h and e_t in KG $\mathcal{G}$.

In this assignment function, we consider whether two entities can be connected via a relation path, instead of the entities or number of the connection paths. There are two main reasons for this design: (1) We expect that the mined causal relationship can be generalized to any dataset in this domain. Thus, if we want to distinguish different entities which instantiate the meta structure, we need to build a multinomial model for all possible entities. The multinomial would be infeasibly large. And our model can not be applied to any scenario which contain an unseen entity. (2) this function can be seen as an aggregation function to summary the connection information between entities. The aggregation function is very common in the causal relation model [25, 20, 21, 35]. With the aggregation function, we can build a concise and expressive model. Since the only thing we need is whether the entities are connected. Based on this assignment function, by sampling entity pairs in the training KG and querying the corresponding variable values, we can obtain tabular data for causal analysis.

3.2 Causal MetaKnowlege Discovery via d-seperation Criterion

The d-separation criteria [13] (see Definition 3.4) is a sufficient and necessary condition for the compatibility of a probability distribution with a causal model in the form of a directed acyclic graph (DAG). It states that a joint probability distribution of a set of random variables is compatible with the DAG (each node represents one of the given variables and each arrow represents the possibility of causal influence) if and only if the distribution satisfies a set of conditional independence relations encoded in the structure of the DAG. Therefore, d-separation is widely used in the algorithms in discovering causal structure[12, 36, 11].

Definition 3.3: d-seperation. A path p is blocked by a set of nodes Z if and only if:

1.p contains a chain of nodes $A \to B \to C$ or a fork $A \leftarrow B \to C$ such that the middle node B is in Z (i.e., B is conditioned on), or:

2. p contains a collider $A \to B \leftarrow C$ such that the collision node B is not in Z, and no descendant of B is in Z.

If Z blocks every path between two nodes X and Y, then X and Y are d-separated, conditional on Z, and thus are independent conditional on Z.

Algorithm 1: Local causal metaknowledge discovery

Input: Y and $\{y_i\}, i = 1, \ldots, N$: variable and samples of queried variable $(C_h, C_t).M_q$;
$\mathcal{X}^{Ca} = \{X_k\}, k = 1, \ldots, K$ and $\{\{x_i\}_k\}, i = 1, \ldots, N$: variables and samples of candidate causes;

Output: causes $\mathcal{X}^C$ of Y

level $d \leftarrow 0$;
while $d <= |\mathcal{X}^{Ca}| - 1$ **do**
  for each $X_k \in \mathcal{X}^{Ca}$ **do**
    for each subset $\mathcal{Z} \in \mathcal{X}^{Ca} \backslash \{X_k\}$ and $|\mathcal{Z}| = d$ **do**
      Test CI($X_k, Y|\mathcal{Z}$);
      if CI($X_k, Y|\mathcal{Z}$) **then**
        Test CI($\mathcal{Z}, Y|X_k$) (Reverse CI test.) ;
        if not CI($\mathcal{Z}, Y|X_k$) **then**
          Remove X_k from $\mathcal{X}^{Ca}$;
          Break;
        end
      end
    end
  end
  $d \leftarrow d + 1$;
end
$\mathcal{X}^C = \mathcal{X}^{Ca}$

In this work, we design an efficient causal metaknowledge discovery algorithm based on d-separation. With d-separation, we can get the following conclusion: given any set of variables Z, where Z does not include X, X is not independent of its parent node (i.e. direct cause). Based on this conclusion, we can obtain a criterion for determining the direct cause of variable X. Furthermore, we design the following local causal metaknowledge discovery algorithm (Algo. **??**) for the queried variable $Y = f_{\mathcal{G}}(x, y|R_h)$. We only mine the direct cause of Y, instead of the entire causal structure of variable set $\mathcal{X}^{Ca} \cup \{Y\}$. Particularly, given a queried variable $Y = f_{\mathcal{G}}(x, y|R_h)$, for each candidate cause in $\mathcal{X}^{Ca}$ (denoted as variable X_k), the proposed algorithm

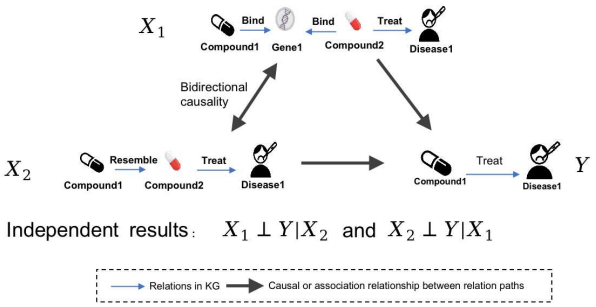


Figure 4: An example of bidirectional causal relationship, which may lead wrong results.

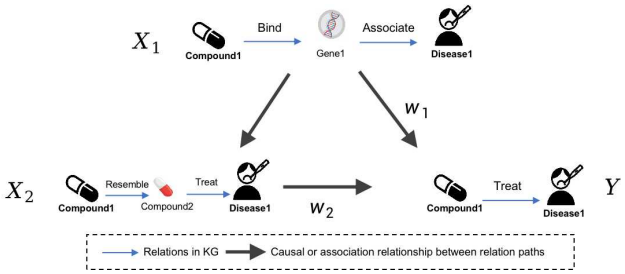


Figure 5: An example of non-independent rule-induced variables, which are both the causes of queried relation.

decides whether X_j should be retained in candidate causes set $\mathcal{X}^{Ca}$ by testing the independence of X_k and Y conditioning on a subset $\mathcal{Z}$ of $\mathcal{X}^{Ca} \backslash \{X_k\}$. The conditional independent(CI) tests are organised by levels (based on the size d of the conditioning sets). At the first level ($d = 0$), all pairs of variables are tested conditioning on the empty set. Some of the candidate causes would be removed and the algorithm only tests the remaining candidate causes in the next level ($d = 1$). The size of the conditioning set, d, is progressively increased (by one) at each new level until d is greater than $|\mathcal{X}^{Ca}| - 1$. Each corresponding relation path of $X \in \mathcal{X}^{C}$ construct a valid rule to predict the relation R_h in Y.

It is noteworthy that we add the reverse CI test in Algo. **??** (line 7) to avoid the impact of redundant relations in KGs. For example, $Compound1 \overset{Resembles}{\longrightarrow} Compound2$ and $Compound1 \overset{Binds}{\longrightarrow} Gene1 \overset{Binds}{\longleftarrow} Compound2$ express the similar message, which could lead the invalid independence test, as shown in Fig. 4. It will lead both X_1 and X_2 are removed from the candidate cause set of queried variable Y, even though they have very strong causal relationship with the drug treatment of diseases. Consequently, we use the reverse CI test to avoid this issue. In particular, if X_j and Y are judged to be independent conditioning on $\mathcal{Z}$, we will examine the independence between $\mathcal{Z}$ and Y conditioning on $\mathcal{X}_j$. When the result of the additional test is negative, X_j will be removed from $\mathcal{X}^{Ca}$. In this paper, we adopt SCI method [26] as the independent test method in the experiments, which works well on limited samples and discrete variables.

3.3 Link Prediction based on Explainable Causal Metaknowledge

The approach for link prediction based on interpretable rules tends to generate corresponding weights in the rule mining phase. By accumulating the weights of the rules satisfied by each predicted entity, a score of the predicted entities can be generated, and then the results are ranked based on this score. Here we first introduce how to generate rule weights under the causal model and then describe the approach for link prediction based on generated weights.

Weights of rules based on conditional dependency. In Algo. **??**, we discover the direct causes by the non-independence relationship between the candidate meta structures and the queried meta structures. It is important

to note that the meta structures of $\mathcal{X}^C$ are not independent to each other. Fig. 5 gives an example for this case. Specifically, X_1 and X_2 are both causes of Y. Since X_1 is also a cause of X_2, if we directly calculate the causal strength between X_2 and Y, it is inevitable that w_2 will contain the causal effects that arise from X_1 along the path $X_1 \rightarrow X_2 \rightarrow Y$. Therefore, in order to better measure the importance of each causal rule and to avoid double-counted in the calculation of each proposed entity's score, we adopt the minimal conditional dependence as a measure of the importance of causal rules:

$$\begin{aligned} w_j = \min(\{dependence(X_j, Y|\mathcal{Z})\}) \\ \text{for any subset } \mathcal{Z} \in \mathcal{X}^{Ca} \backslash \{X_j\}, \end{aligned} \tag{15}$$

where w_j is the rule weight of the meta structure in X_j. In this paper, we use the $SCI_f(X, Y|Z)$ in SCI independence test [26] as the dependence score in Eq. 15, which can be get in the process of causal rules discovery. The higher of $SCI_f(X, Y|Z)$, the stronger the dependency.

Score function of entity results. Because of the incompleteness nature of KGs, open world assumption (OWA) [17] is often considered on real datasets. Under the OWA, the SUM function are usually adopted to calculate the ranking score of the predicted entity e_h in link prediction task $(?, R_h, e_t)$:

$$S_{R_q}^{sum} = \sum_{i=1}^{K} \tilde{w}_i Q_i, . \tag{16}$$

where K is the number of causal rules, $\tilde{w}_i$ is the normalized weight. $Q_i = 1$ when the body of the i-th causal rule holds for the entity pair (e_h, e_t), otherwise $Q_i = 0$. This approach focuses on the entities supported by multiple rules and does not use the non-existent relations between entity pairs, since the unreliable negative samples under OWA. In this paper, Eq. 16 is used in the link predictions on real data. For KG under closed world assumption(CWA) [17], the negative facts are also reliable, therefore we design a new function to apply the rules in the link prediction task. Particularly, given an query $(?, R_h, e_t)$, the score of the triple (e_h, R_h, e_t) is true can be formulated as:

$$S_{R_q}^{avg} = \sum_{i}^{K} \tilde{w}_i \big(Q_i \bar{Y}_{X_i=1} + (1 - Q_i)\bar{Y}_{X_i=0}\big), \tag{17}$$

where K is the number of causal rules for the queried relation, $\tilde{w}_i$ is the normalized weight for the i-th result rule. $\bar{Y}_{X_i=1}$ denotes the proportion of the queried relation to be true when the body of the i-th causal rule is true in the training data, and $\bar{Y}_{X_i=0}$ denotes the proportion of the queried relation to be true when the body of the i-th causal rule is false. $Q_i = 1$ when the body of the i-th causal rule holds for the entity pair (e_h, e_t), otherwise $Q_i = 0$. The results will be ranked by S_{R_q} of each valid e_t. In this paper, Eq. 17 is used in the link predictions on simulation data.

4 Experimental Study

4.1 Experimental Setup

In this section, we empirically evaluate the effectiveness and interpretability of the proposed CMLP on both simulation and real-world datasets. For interpretability, we focus on whether the algorithm can uncover the causal relationships inherent in the knowledge graph.

4.1.1 Baselines

To evaluate the interpretability of the algorithms, we select four rule-based methods that can conduct link prediction and generate explainable rules. To make a fair comparison, the inference rules, obtained from different

Table 11: Dataset statistics of all the experiments.

	#Triplets	#Relations	#Entities
Simulation	6,095	5	1,590
Douban Movie Rate	28,356	12	3,007
Hetionet	174,941	20	32,056

algorithms, are used to conduct the link prediction task based on the same prediction equations. This approach can also help us observe the impact of different rules on the link prediction task. For a complete evaluation of the effectiveness of the proposed approach, we also compute the LP performance of TuckER[2], one representation-based method which has the best overall performance among the representation-based methods across different datasets[32]. All baselines are listed in the following:

1) AMIE+[8], an efficient top-down method to discover the interpretable rules.

2) AnyBURL[27], a bottom-up approach to mine the logical rule.

3) Neural-LP[44], an end-to-end differentiable model to learn the first-order logical rule.

4) RNNLogic[30], an EM-based algorithm to learn the rule generator and the reasoning predictor iteratively.

5) TuckER[2], a linear model based on Tucker decomposition of the binary tensor representation of knowledge graph triples.

4.1.2 Datasets

To quantitatively evaluate the effectiveness of the algorithm in discovering causal knowledge, we construct a simulation dataset owing to a lack of groundtruth of real datasets. Douban and Hetionet [15] are selected as our real datasets on which we perform two link prediction tasks, movie rating prediction and drug repurposing, respectively. Here we provide more details for these datasets, and their statistics are shown in Table 11.

Simulation dataset. We generate simulated KGs based on a toy causal model specified in Fig. 6, which includes three concepts and five relations. In particular, we design the causal mechanisms in KG via a probabilistic model. The root nodes (X_1, X_4) in the causal graph are generated via Bernoulli distributions, whose probability mass function is $f_X(x) = p^x(1-p)^{1-x}$. Moreover, the non-root nodes (X_2, X_3) are generated via the conditional probability distributions, which are Bernoulli distributions, given the parent node (X_1). To maintain a stable causal mechanism, the parameters of conditional distributions are constant in training and testing, as shown in Table 12. In the out-of-distribution paragraph, we will introduce the parameters of root nodes in training and testing.

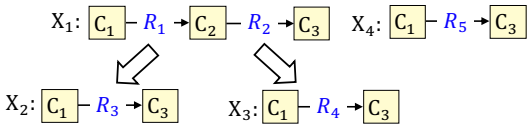


Figure 6: The causal graph of relation paths, based on which the simulated KGs are generated .

Douban movie rating. Douban is a famous Chinese website for movie reviews, where users can rate and comment on any movie. The rating range is from 1 to 5. A higher rating means that users like movies, while a lower rating means that users have negative feedback on movies. We collect the real-world data from Douban[1]

[1]https://www.douban.com/

Table 12: The parameters of conditional distributions.

Conditions	$X_2\|X_1=1$	$X_2\|X_1=0$	$X_3\|X_1=1$	$X_3\|X_1=0$
Parameters	p=0.9	p=0.1	p=0.9	p=0.1

and construct a dataset (this dataset will be released), whose statistics are shown in Table 11. Commonly, a movie with a score of 4 or 5 is identified as meeting the taste of users. So we transform the original 5-level rating to a 2-level rating with a threshold of 4. If the rating score is 4 or 5, the original relation *Rate* is replaced by *HighRate* We conduct the link prediction task on the relation *HighRate*. Because the raw data is too large and the relations between users and movies are very sparse, in this thesis, we first filter the 20 users who have made the most ratings and take the rating history of these users as the set of rating facts for our study. The facts unrelated to the rating are also included in our experimental data.

Hetionet[15] is a freely available knowledge database that integrates biomedical information from 29 prominent bioinformatics resources. Recently, Hetionet was successfully applied to drug repurposing tasks in terms of the link prediction task for relation `Treats`[15, 31].

Why do we choose those two real datasets instead of other commonly used datasets, such as WN18, FB15k? In this paper, we focus on link prediction with the help of causal relationships between knowledge graph relations. The core of causality lies in its asymmetry. The commonly used KGs for link prediction algorithms contain many symmetrical relationships, e,g., *hypernym* and *hyponym* in WN18. These symmetrical relationships may help with the link prediction task, but they go against the basic idea that causality is a one-way relationship. We, therefore, chose datasets with specific application scenarios and rich causal semantics.

4.1.3 Metrics.

For link prediction, we employ the commonly used metrics mean reciprocal rank (MRR) and Hits@k [32, 2, 3].

$$MRR = \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{q} \tag{18}$$

$$Hit@K = \frac{|\{q \in Q : q \leq K\}|}{|Q|} \tag{19}$$

where Q is the rank results, for each $q_i \in Q$ is the ranking of the desired results in the i-th query $(?, R_h, e_t)$. In the case of ties in the calculation of Q, we use the mean rank to avoid misleading results[32, 33]. Both MRR and His@k are the higher, the better.

The interpretability of causal rules on the simulation dataset can be understood as the consistency between the mined causal rules and the actual causal structure. Therefore, we evaluate baselines and our approach on the simulation dataset using precision, recall, and structural Hamming distance (SHD) as the evaluation metrics, which are commonly used evaluation metrics in causal structure discovery studies [46, 47].

$$Precision = \frac{\#TFR}{\#FR}; Recall = \frac{\#TFR}{\#ATR} \tag{20}$$

Where $\#TFR$ is the number of right causal relationships discovered by an algorithm, $\#FR$ is the number of causal relationships discovered by an algorithm, and $\#ATR$ is the number of all causal relationships. Both precision and recall are the higher, the better. SHD calculates the difference between the learned graph and the ground truth graph by the number of edge insertions, deletions, or flips required to transform one graph into another. The lower the SHD, the better.

4.1.4 Out-of-Distribution Link Prediction

Traditional machine learning methods are designed based on the assumption of independent and identically distributed (I.I.D) data. This assumption means the training and test data come from the same distribution. However, the distribution of test data may alter due to changes in the test environment; such tasks are referred to as OoD tasks. Traditional algorithms perform poorly on generalization problems due to the violation of I.I.D assumption. Causality is seen as a stable inference mechanism in many research works on generalization problems. So, in this work, we provide an out-of-distribution generalization task for the knowledge link prediction task for the first time. On the one hand, the effect of causal metaknowledge on this task can be measured, and on the other, the performance of existing algorithms can be looked at. We also evaluate the performance of the algorithms on I.I.D link prediction tasks.

In this paper, we design two OoD experimental scenarios.

(1) *Simulation dataset*: We evaluate the link prediction performance under I.I.D and OoD settings, where the triples of root node X_1 are generated in testing under the same and different parameters with training. In the training and I.I.D testing datasets, $p_{X_1} = 0.5$. In the OoD testing datasets, $p_{X_1} \in \{0.2, 0.9\}$. For X_4, p_{X_4} maintains 0.9 in training and testing. The facts of the KGs are split into three parts:*train*, *test info*, and *test*. The facts in *train* are used to learn the rule. The effectiveness of the learned rule is assessed via the link prediction task on R_3. The *test info* part includes facts of new entities (did not appear in *train*) on R_1, R_2, R_4, R_5, and *test* part includes the queried facts on R_3.

(2)*Real datasets*: It is impossible to explicitly change the data distribution since real data distribution is inaccessible for real datasets. Recent research[38, 43] has suggested that graph models are biased towards nodes with larger degrees, which causes the bad performance of low-degree nodes in the test. Therefore we construct the OoD datasets based on degree shift. Specifically, given a query task $(?, R_h, e_t)$, we calculate the *median* of degree[2] of known entities e_t belonging to the triples (e_h, R_h, e_t) in the training. Then we bin the test queries by the degrees of the known entities e_t. The degree range in each bucket is decided based on the sample size balance. The test queries in the bucket, which the training median falls in, can be treated as the I.I.D test samples. Others are the OoD samples. The I.I.D bucket is labeled with $*$ in Table 14 and Table 15.

4.2 Performance of Link prediction task in Out-of-Distrition Settings

Results on simulations. For the simulation dataset, we construct a OoD setting named as covariance shift, by changing the probability distribution of the root nodes in the test phase. Table 13 presents the methods in performance and demonstrates the effectiveness of the proposed CMLP. In particular, CMLP outperforms the baseline models under all metrics except the Hits@10 under I.I.D setting. This shows our method can give a stable and high-quality result, especially in the OOD setting. Besides, compared with the baselines, the proposed CMLP perform significantly better at the Hits@1 metric (at least 25% absolute improvements that the second place under all settings), which suggests the our method are more suitable for scenarios with strict performance requirements, and this feature may achieved by the removal of association-based rules.

Results for Douban movie rating. Since we filtered the users of the Douban data, and the discrepancies of the degrees of experimental user nodes are close to each other. Therefore, in this experiment, to construct the OoD scenario, we adopt the head prediction (? , HighRate, Movie), predicting the set of users who gave high ratings to movies. Further, we bucketed the movie nodes in the test data according to their degree in the training data to observe the performance of the algorithm under different node prevalence. The MRR and Hits@5 results shown in Tab. 14 shows that the proposed CMLP get the best performance in the all OoD settings. Especially, at least 25.8% and 29.3 % relative improvements that the second place on the MRR and Hits@5, respectively. In the I.I.D setting, CMLP gets the second place on both MRR and Hits@5, lower than the representation-based method

[2]In this paper, we use the term "degree" to stand for the sum of in and out degrees

Table 13: The results of link prediction on simulation datasets.

Settings	p_{X_1}	Method	MRR	Hits		
				@10	@3	@1
I.I.D	0.5	AMIE+	0.87	**98.99**	94.95	78.79
		AnyBURL	0.87	**98.99**	94.95	78.79
		Neural-LP	0.80	**98.99**	92.42	66.16
		RNNLogic	0.87	**98.99**	94.95	78.79
		CMLP	**0.94**	98.48	**97.98**	**90.91**
OOD	0.2	AMIE+	0.875	96.91	93.81	81.44
		AnyBURL	0.875	96.91	93.81	81.44
		Neural-LP	0.68	98.97	79.38	50.51
		RNNLogic	0.875	96.91	93.81	81.44
		CMLP	**0.99**	**100**	**100**	**99.97**
OOD	0.9	AMIE+	0.91	**100**	96.34	85.67
		AnyBURL	0.91	**100**	96.34	85.67
		Neural-LP	0.88	**100**	96.95	79.57
		RNNLogic	0.91	**100**	96.34	85.67
		CMLP	**0.99**	**100**	**99.70**	**99.09**

TuckER. These results illustrate that for movie rating datasets, the rules learned by our method can capture more general user preferences and give relatively accurate rating predictions for movies that are in different popularity.
Results for drug repurposing on Hetionet. Consistent with the traditional setup of drug redirection, on Hetionet, we also use head predition (?, Treat, Disease), which is giving a Disease to predict new drugs. We also observe the performance of the algorithm under this task by bucketing for Disease node degrees. Tab. 15 reports the MRR and Hits@5 on this dataset, and we can find that: our CMLP performs significantly better than other baselines on low-degree diseases (0-17), while AMIE+ and AnyBURL get better results on low-degree diseases (17-100). These results indicate that our method can give more accurate drug discovery results for diseases with relatively low information. And for diseases with richer information, the correlation-based inference rules give more accurate drug prediction.

4.3 Quality and Interpretability of Causal Rules.

As stated in Sec. 1, the mined rules play a key role in our algorithm, and an important advantage of these rules is that they are well interpretable. In this section, we will evaluate the quality and interpretability of rules mined by CMLP .
Quality of causal rules from simulations. The ground-truth causal graph of KGSs shown in Fig 6, and Table 16 shows the accuracy of estimated rules of different methods. In particular, CMLP accurately discovers two causal rules in Fig 6 without any redundant rules. In contrast, correlation-based methods report some non-causal rules.
Interpretability of causal rules Moreover, for the simulation dataset, we analyze all methods' results, whose heads are R_3 and R_5, and the results are shown in Table 17 (We omit results of R_4, since R_3 and R_4 are symmetric in the causal graph). The rules follow the causal mechanism are in bold. There is only one causal rule for R_3, which is $R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)$. AMIE+, AnyBURL, RNNLogic and CMLP find this causal rule. Besides this causal rule, the results of AMIE+, RNNLogic and AnyBURL also include other rules, such as

Table 14: MRR (left) and Hits@5 (right) for Douban movie rating. The * marks columns that contain the I.I.D results. Other columns contain OoD results.

Methods	Degree Range			
	0-21*	**21-31**	**31-39**	**39-60**
AMIE+	0.120	0.205	0.261	0.395
AnyBURL	0.125	0.182	0.231	0.373
Neural-LP	0.078	0.097	0.126	0.217
RNNLogic	0.072	0.086	0.097	0.161
TuckER	**0.287**	0.186	0.186	0.149
CMLP	0.251	**0.343**	**0.392**	**0.497**

Methods	Degree Range			
	0-21*	**21-31**	**31-39**	**39-60**
AMIE+	13.7	30.7	39.4	68.3
AnyBURL	15.1	26.0	33.2	64.6
Neural-LP	0.1	0.6	3.4	60.0
RNNLogic	1.6	4.7	7.2	13.5
TuckER	**50.3**	31.9	28.2	21.5
CMLP	40.9	**60.0**	**68.1**	**88.3**

Table 15: MRR(left) and Hits@5(right) for drug repurposing on Hetionet. The * marks columns that contain the I.I.D results. Other columns contain OoD results.

Methods	Degree Range			
	0-8*	**8-17**	**17-31**	**31-100**
AMIE+	0.103	0.085	**0.132**	0.065
AnyBURL	0.116	0.189	0.090	**0.188**
Neural-LP	0.027	0.014	0.009	0.009
RNNLogic	0.07	0.021	0.029	0.012
TuckER	0.044	0.022	0.083	0.015
CMLP	**0.248**	**0.208**	0.095	0.093

Methods	Degree Range			
	0-8*	**8-17**	**17-31**	**31-100**
AMIE+	13.2	11.3	**22.5**	13.2
AnyBURL	15.8	**25.0**	9.6	**23.7**
Neural-LP	5.3	0	0	0
RNNLogic	7.9	0	3.2	0
TuckER	3.1	5.9	10.7	3.2
CMLP	**26.31**	**25.0**	16.1	13.1

Table 16: Experimental results on simulation data with $p_{X_1} = 0.5$, based on the metrics (precision, recall and SHD), which are commonly used to evaluate the estimated causal graph.

Method	**Precision ↑**	**Recall ↑**	**SHD ↓**
Neural-LP	0	0	10
AMIE+	0.22	**1.0**	7
RNNLogic	0.22	**1.0**	7
AnyBURL	0.25	**1.0**	6
CMLP	**1.0**	**1.0**	**0**

Table 17: All rules whose head are R_3 and R_5, obtained by each algorithm learned on simulated dataset. The strikethroughs indicate the wrong results (there is no entities satisfying the rule). The rules consistent with the generation process are in bold. The orange text denotes the weight of each rule with the form max-normalization(original weight)

.

Method	**Rules of R_3 with $\mathbf{p_{X_1} = 0.5}$**	**Rules of R_3 with $\mathbf{p_{X_1} = 0.9}$**	**Rules of R_5**
AMIE+	1.00 (0.908) $\mathbf{R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)}$ 0.91 (0.829) $R_3(C_1, C_3) \leftarrow R_4(C_1, C_3)$ 0.55 (0.500) $R_3(C_1, C_3) \leftarrow R_5(C_1, C_3)$	1.00 (0.900) $\mathbf{R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)}$ 0.99 (0.890) $R_3(C_1, C_3) \leftarrow R_4(C_1, C_3)$ 0.91 (0.820) $R_3(C_1, C_3) \leftarrow R_5(C_1, C_3)$	1.00 (0.898) $R_5(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)$ 0.99 (0.895) $R_5(C_1, C_3) \leftarrow R_3(C_1, C_3)$ 0.99 (0.894) $R_5(C_1, C_3) \leftarrow R_4(C_1, C_3)$
AnyBURL	1.00 (0.896) $\mathbf{R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)}$ 0.92 (0.823) $R_3(C_1, C_3) \leftarrow R_4(C_1, C_3)$ 0.56 (0.501) $R_3(C_1, C_3) \leftarrow R_5(C_1, C_3)$	1.00 (0.898) $R_3(C_1, C_3) \leftarrow R_4(C_1, C_3)$ 0.99 (0.893) $\mathbf{R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)}$ 0.91 (0.821) $R_3(C_1, C_3) \leftarrow R_5(C_1, C_3)$	1.00 (0.907) $R_5(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)$ 0.99 (0.902) $R_5(C_1, C_3) \leftarrow R_3(C_1, C_3)$ 0.99 (0.897) $R_5(C_1, C_3) \leftarrow R_4(C_1, C_3)$
Neural-LP	1.00 (0.318) ~~$R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_4(C_2, C_3)$~~ 0.99 (0.316) ~~$R_3(C_1, C_3) \leftarrow R_4(C_1, C_2), R_5(C_2, C_3)$~~ 0.31 (0.100) ~~$R_3(C_1, C_3) \leftarrow R_5(C_1, C_2), R_4(C_2, C_3)$~~ 0.31 (0.099) ~~$R_3(C_1, C_3) \leftarrow R_5(C_1, C_2), R_5(C_2, C_3)$~~ 0.23 (0.073) $R_3(C_1, C_3) \leftarrow R_4(C_1, C_3)$ 0.09 (0.028) ~~$R_3(C_1, C_3) \leftarrow R_4(C_1, C_2), R_4(C_2, C_3)$~~ 0.07 (0.023) $R_3(C_1, C_3) \leftarrow R_5(C_1, C_3)$	1.00 (0.757) ~~$R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_1(C_2, C_3)$~~ 0.17 (0.128) ~~$R_3(C_1, C_3) \leftarrow R_1(C_1, C_3)$~~ 0.04 (0.056) ~~$R_3(C_1, C_3) \leftarrow R_5(C_2, C_3), R_1(C_1, C_2)$~~ 0.05 (0.035) ~~$R_3(C_1, C_3) \leftarrow R_4(C_2, C_1), R_1(C_2, C_2)$~~ 0.03 (0.025) $\mathbf{R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)}$	1.00 (0.125) ~~$R_5(C_1, C_3) \leftarrow R_4(C_1, C_2), R_3(C_2, C_3)$~~ 0.78 (0.097) ~~$R_5(C_1, C_3) \leftarrow R_3(C_1, C_2), R_3(C_2, C_3)$~~
RNNLogic	1.00 (0.076) $\mathbf{R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)}$ 0.58 (0.044) $R_3(C_1, C_3) \leftarrow R_4(C_1, C_3)$ 0.13 (0.010) $R_3(C_1, C_3) \leftarrow R_5(C_1, C_3)$	1.00 (0.071) $\mathbf{R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)}$ 0.49 (0.035) $R_3(C_1, C_3) \leftarrow R_4(C_1, C_3)$ 0.14 (0.010) $R_3(C_1, C_3) \leftarrow R_5(C_1, C_3)$	1.00 (0.220) $R_5(C_1, C_3) \leftarrow R_3(C_1, C_3)$ 0.28 (0.060) $R_5(C_1, C_3) \leftarrow R_4(C_1, C_3)$ 0.20 (0.045) $R_5(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)$
CMLP	1.00 (122.797) $\mathbf{R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)}$	1.00 (20.061) $\mathbf{R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)}$	-

$R_3(C_1, C_3) \leftarrow R_4(C_1, C_3)$ and $R_3(C_1, C_3) \leftarrow R_5(C_1, C_3)$. Especially, for those three methods, note that the weights of $R_3(C_1, C_3) \leftarrow R_4(C_1, C_3)$ are very close to the weights of $R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_2(C_2, C_3)$. It means the algorithms think these two rules have the similar interpretability for the head relation R_3. With the change of root node X_1's distribution(from $p_{X_1} = 0.5$ to $p_{X_1} = 0.9$), AnyBURL even report the higher weight for the rule $R_3(C_1, C_3) \leftarrow R_5(C_1, C_3)$. According to the generation mechanism, the existence of R_3 between entities e_i and e_j is independent with whether there is R_4 and R_5 between e_i and e_j. AMIE+, AnyBURL and RNNLogic still return these association rules with high weights, because they only consider whether R_3 and R_4 co-occur frequently, but not the reason of the co-occurrence. The end-to-end completion-oriented method, Neural-LP, also return some wrong results, such as the top 1 rule $R_3(C_1, C_3) \leftarrow R_4(C_1, C_2), R_4(C_2, C_3)$, which can not be satisfied by any entities in KG. The results in [34] show the same phenomenon. The intermediate results of the completion-oriented method is incomprehensible sometimes.

Furthermore, We sort the rules generated by each algorithm based on their assigned weights and show the five top rules from Douban and Hetionet in Tab. 18 and Tab. 19, respectively. The results in Tab. 18 suggests that the ratings for the target movie are highly related to other movies which share the same staff, such as writer, actors, director, etc. According to the rating results, CMLP finds a strong causal relationship between the rating of the movie and its editor than other pairs. The top rules generated by AMIE+ and AnyBURL focus on other shared staffs, but the shared staff has different roles in the target movie and the movie in path. Those rules of AMIE+ and AnyBURL are hard to be satisfied for most queries. It is worth noting that RNNlogic report the 'fan' rules will impact the users' rating, but CMLP excludes this kind of rules. Our results suggest the working ability of the movie's stuff (*e.g.* actor or writer) should be the root cause of the users' rate, instead of the followers of the stuff. From the rules from Hetionet, we can see the learned rules are broadly divided into two classes, those in which the target drug and disease are connected by therapeutic information about the similar disease and drug, and those in which the target drug and disease are connected by commonly associated genes. Further, we find that the rules

Table 18: Top 5 Rules to infer HighRate(User, Movie) given by the methods. The strikethroughs indicate the wrong results (there is no entities satisfying the rule).

Method	Top rules to infer HighRate(User, Movie)
AMIE+	1.00 (0.565) HighRate(User,Movie) ← HighRate(User,Movie1), Writer(Person,Movie1),Director(Person,Movie) 0.98 (0.556) HighRate(User,Movie) ← HighRate(User,Movie1), Director(Person,Movie1), Writer(Person,Movie) 0.87 (0.489) HighRate(User,Movie) ← HighRate(User,Movie1), Writer(Person,Movie1), Actress(Person,Movie) 0.74 (0.417) HighRate(User,Movie) ← HighRate(User,Movie1), Director(Person,Movie1), Actor(Person,Movie) 0.72 (0.405) HighRate(User,Movie) ← HighRate(User,Movie1), Actress(Person,Movie1), Writer(Person,Movie)
AnyBURL	1.00 (0.400) HighRate(User,Movie) ← HighRate(User,Movie1), Composer(Person,Movie1), Actor(Person,Movie) 0.99(0.397) HighRate(User,Movie) ← HighRate(User,Movie1), Producer(Person,Movie1), Director(Person,Movie) 0.97 (0.386) HighRate(User,Movie) ← HighRate(User,Movie1), Director(Person,Movie1), Actress(Person,Movie) 0.89 (0.355) HighRate(User,Movie) ← HighRate(User,Movie1), Writer(Person,Movie1), Actress(Person,Movie) 0.85 (0.340) HighRate(User,Movie) ← HighRate(User,Movie1), Editor(Person,Movie1), Editor(Person,Movie)
Neural-LP	1.00 (0.120) HighRate(User,Movie) ← HighRate(User,Movie1), HighRate(User1,Movie1), HighRate(User1,Movie) 0.28 (0.034) HighRate(User,Movie) ← HighRate(User,Movie1), MovieType(Movie1,Type), MovieType(Movie,Type)
RNNLogic	1.00 (0.011) HighRate(User,Movie) ← Fan(User,Person),Editor(Person,Movie) 0.45 (0.005) HighRate(User,Movie) ← Fan(User,Person),Actor(Person,Movie) 0.36 (0.004) HighRate(User,Movie) ← Fan(User,Person),Director(Person,Movie) 0.36 (0.004) HighRate(User,Movie) ← Fan(User,Person),Writer(Person,Movie) 0.36 (0.004) HighRate(User,Movie) ← Fan(User,Person),Composer(Person,Movie)
CMLP	1.00 (0.034) HighRate(User,Movie) ← HighRate(User,Movie1), Editor(Person,Movie1), Editor(Person,Movie) 0.12 (0.004) HighRate(User,Movie1) ← HighRate(User,Movie1),Cinematographer(Person,Movie),Cinematographer(Person,Movie) 0.06 (0.002) HighRate(User,Movie1) ← HighRate(User,Movie1),Writer(Person,Movie),Writer(Person,Movie) 0.06 (0.002) HighRate(User,Movie1) ← HighRate(User,Movie1),Actress(Person,Movie),Actress(Person,Movie) 0.03 (0.001) HighRate(User,Movie1) ← HighRate(User,Movie1),Director(Person,Movie),Actor(Person,Movie)

Table 19: Top 5 Rules to infer Treats(Compound, Disease) given by the methods. For brevity, we use 'C' and 'D' for compound and disease, respectively.

Method	Top rules to infer Treats(Compound, Disease)
AMIE+	1.00 (0.393) Treats(C, D) ← Resembles(C,C1), Treats(C1, D) 0.82 (0.322) Treats(C, D) ← Resembles(C1,C),Treats(C1, D) 0.42 (0.167) Treats(C, D) ← Downregulates(C,Gene1), Associates(D,Gene1) 0.38 (0.151) Treats(C, D) ← Downregulates(C,Gene1), Upregulates(D,Gene1) 0.37 (0.144) Treats(C, D) ← Binds(C,Gene1), Upregulates(D,Gene1)
AnyBURL	1.00 (0.319) Treats(C, D) ← Includes(PharmacologicClass1,C),Includes(PharmacologicClass1,C1),Treats(C1, D) 0.60 (0.192) Treats(C, D) ← Resembles(C1,C),Treats(C1, D) 0.52 (0.166) Treats(C, D) ← Resembles(C1,C),Resembles(C1,C2),Treats(C2, D) 0.31 (0.098) Treats(C, D) ← Resembles(C1,C),Resembles(C2,C1),Treats(C2, D) 0.24 (0.077) Treats(C, D) ← Treats(C, D1), Resembles(D1,D)
Neural-LP	1.00 (0.659) Treats(C, D) ← Treats(C, D1), Treats(C1, D1),Treats(C1, D)
RNNLogic	1.00 (0.00007) Treats(C, D) ← Resembles(C, C1),Treats(C1, D) 1.00 (0.00007) Treats(C, D) ← Resembles(C, C1), Resembles(C1, C2),Treats(C2, D)
CMLP	1.00 (269.00)) Treats(C, D) ← Treats(C, D1),Resembles(D1, D2),Resembles(D, D2) 0.85 (229.32)) Treats(C, D) ← Includes(PharmacologicClass1, C),Includes(PharmacologicClass1, C1),Treats(C1, D) 0.83 (224.37)) Treats(C, D) ← Treats(C, D1),Resembles(D2, D1),Resembles(D2, D) 0.58 (155.18)) Treats(C, D) ← Treats(C, D1),Treats(C1, D1),Treats(C1, D) 0.10 (26.52)) Treats(C, D) ← Treats(C, D1), Resembles(D2,D1), Resembles(D,D1)

mined by AnyBURL also contain rules for reasoning through shared side effects.

5 Related Work

In this section, we first review the related studies in causal discovery for propositional domains and relational domains. Then we discuss and clarify the distinction between the proposed approach and the most relevant *rule mining* methods for relational data. We list the relevant research areas and our differences in the Tab. 20

Table 20: Comparison of our work and related work.

	Association	Causality
Propositional	Traditional machine learning	Traditioanal causal model
relational	Rule mining(e.g.logical rule), Graph representation learning	Relational causal model(with attribute), **Our(without attribute)**

Causal Discovery from Propositional Data. Rubin causal models [16] and structural causal models (SCMs) [29] are the two dominated frameworks for causal discovery from propositional data. Particularly, the former analyzes the causal effect between treatment and effect with partial structural information, while the later employs Bayesian

networks to identify causal structure. Our situation resembles causal discovery in SCM because we are primarily concerned with unearthing causal relationships from KG. Furthermore, there are two kinds of causal discovery algorithms, *constraint-based* and *score-based* [37], in SCMs. Contrary to the score-based approaches, which are based on the global score, the constraint-based approaches can employ the local conditional independence to determine the causal relationship between specific variables, which is critical when only partial causal relationship is interested. In addition, the constraint-based methods are non-parametric, which means that they do not depend on the specific functions to connect variables. Based on above advantages, we follow the constrained-based design to develop our method.

Relational Causal Model. To our best knowledge, the relational causal model [25, 20, 21, 35] is the only framework designed to extract causal information from relational data. The input of a relational causal model is a relational database containing entity and relation tables. Each entity table contains all entities corresponding to the same concept, and each relation table contains all facts corresponding to the same relationship between two entities. Consequently, a relational database is comparable to a KG. Relational causal model finds causal relationship between related entity attributes. For example, for the database with two relations: Develop(Employee, Product) and Funds(Company, Product), the relational causal relationship will give the results like [Employee, Develops, Products, Fundsby, Company].budgets $\rightarrow$ [Employee].Success. We can see that relational causal models emphasize relational models of attributes with a known entity-level connection graph [24], while our research focuses on generative process of the connection, which is the preceding step in the entire KG generation process.

Rule Mining from KG. Inductively knowledge reasoning involves generalising patterns from a given set of observed facts and then generating novel but potentially imprecise predictions. In addition to the incomprehensible techniques based on embedding, *rule mining* methods, which benefit from intuitive interpretation of the findings of link prediction, have maintained the popularity for decades. The rule mining studies in KG can be divided into two categories according to the main objectives: metrics-oriented and completion-oriented. Metrics-oriented methods [9, 8, 28] usually use predefined co-occurrence metrics, *confidence* and *support*, to find rules satisfying the given thresholds of the metrics, based on a top-down fashion. Recently, AnyBURL [27] designs a bottom-up technique for rule learning, which requires few computational resources. The other line of research is completion-oriented. Different from the predefined metrics-based methods, these studies are mainly based on end-to-end learning and target on the link completion task. The explainable rules are mainly the intermediate results, which are obtained via analyzing the parameters of the model. The trained models are used to predict the link directly. Neural-LP [44] adopts an attention mechanism to select a variable-length sequence as the body of rules for which confidence scores are learnt from the attention vectors. DRUM [34] uses bidirectional recurrent neural networks to learn the relations of sequences, which are the body of rules, and their confidence scores are estimated via the recurrent neural network. RNNLogic [30] utilizes logic rules as a latent variable and trains both a rule generator and a reasoning predictor with logic rules.

Our work can be seen as one of solutions for rule mining. Different from the past the association-based rule mining methods, our work aims to discover deeper relationship (*i.e.* causation) via a more rigorous statistical inference system. To the best of our knowledge, this is the fist attempt to study rule mining problem under causal perspective, as far as we know.

6 Conclusion and Future work

In this work, we propose a method, CMLP, for entity-level link prediction based on the causal relationships between topologies at the concept level. This method constructs complex connectivity between entities and predicts causal relationships between links at the conceptual level, eliminating spurious correlation that may be learned by traditional association-based methods. Extensive experiments have shown that the proposed CMLP achieves leading performance in a variety of OOD experimental settings. Note that previous representation learning based

models (generally learning from correlations) are hard to generalize to OOD setting, and our model demonstrates that causal-based learning is a promising solution for this setting.

Since the causal model itself is the method which modes the process of data generation, the mined causal rule can be used to understand the physical mechanism of knowledge graph generation. It opens up new possibilities for research in fields such as pharmaceutical economics. Besides, although this paper propose a rule-based model which can works under OOD setting, how to improve the generalization ability of the representation-based models is still a open problem and deserved further investigation.

References

[1] John Aldrich. Correlations Genuine and Spurious in Pearson and Yule. Statistical Science, 10(4):364 – 376, 1995.

[2] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. In Empirical Methods in Natural Language Processing, 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] Xuelu Chen, Ziniu Hu, and Yizhou Sun. Fuzzy logic based logical query answering on knowledge graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pages 3939–3948, 2022.

[5] Xiao Cui and Hao Shi. A*-based pathfinding in modern computer games. International Journal of Computer Science and Network Security, 2011.

[6] James A Evans and Jacob G Foster. Metaknowledge. Science, 2011.

[7] Santo Fortunato, Carl T Bergstrom, Katy Börner, James A Evans, Dirk Helbing, Staša Milojević, Alexander M Petersen, Filippo Radicchi, Roberta Sinatra, Brian Uzzi, et al. Science of science. Science, 2018.

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

[9] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In Proceedings of the 22nd international conference on World Wide Web, pages 413–422, 2013.

[10] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In Proceedings of the 22nd international conference on World Wide Web - WWW '13, pages 413–422. ACM Press.

[11] Andreas Gerhardus and Jakob Runge. High-recall causal discovery for autocorrelated time series with latent confounders. Advances in Neural Information Processing Systems, 33:12615–12625, 2020.

[12] Enrico Giudice, Jack Kuipers, and Giusi Moffa. The dual pc algorithm for structure learning. In International Conference on Probabilistic Graphical Models, pages 301–312. PMLR, 2022.

[13] Madelyn Glymour, Judea Pearl, and Nicholas P Jewell. Causal inference in statistics: A primer. John Wiley & Sons, 2016.

[14] Manuel Heusner, Thomas Keller, and Malte Helmert. Best-case and worst-case behavior of greedy best-first search. International Joint Conferences on Artificial Intelligence, 2018.

[15] Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. Elife, 6:e26726, 2017.

[16] Guido W Imbens and Donald B Rubin. Rubin causal model. In Microeconometrics, pages 229–241. Springer, 2010.

[17] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems, 33(2):494–514, 2021.

[18] Daniel R Lanning, Gregory K Harrell, and Jin Wang. Dijkstra's algorithm and google maps. In Proceedings of the 2014 ACM Southeast Regional Conference, 2014.

[19] Thuc Duy Le, Tao Hoang, Jiuyong Li, Lin Liu, Huawen Liu, and Shu Hu. A fast pc algorithm for high dimensional causal discovery with multi-core pcs. IEEE/ACM transactions on computational biology and bioinformatics, 16(5):1483–1495, 2016.

[20] Sanghack Lee and Vasant Honavar. On learning causal models from relational data. In Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[21] Sanghack Lee and Vasant Honavar. Towards robust relational causal discovery. In Uncertainty in Artificial Intelligence. PMLR, 2020.

[22] Haotian Li, Yong Wang, Songheng Zhang, Yangqiu Song, and Huamin Qu. Kg4vis: A knowledge graph-based approach for visualization recommendation. IEEE Transactions on Visualization and Computer Graphics, 28(1):195–205, 2021.

[23] Zizheng Lin, Haowen Ke, Ngo-Yin Wong, Jiaxin Bai, Yangqiu Song, Huan Zhao, and Junpeng Ye. Multi-relational graph based heterogeneous multi-task learning in community question answering. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pages 1038–1047, 2021.

[24] Marc Maier, Katerina Marazopoulou, David Arbour, and David Jensen. A sound and complete algorithm for learning causal models from relational data. In Uncertainty in Artificial Intelligence, page 371. Citeseer, 2013.

[25] Marc Maier, Brian Taylor, Huseyin Oktay, and David Jensen. Learning causal models of relational domains. In Proceedings of the AAAI Conference on Artificial Intelligence, 2010.

[26] Alexander Marx and Jilles Vreeken. Testing conditional independence on discrete data using stochastic complexity. In The 22nd International Conference on Artificial Intelligence and Statistics, 2019.

[27] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime bottom-up rule learning for knowledge graph completion. In IJCAI, pages 3137–3143, 2019.

[28] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. An embedding-based approach to rule learning in knowledge graphs. IEEE Transactions on Knowledge and Data Engineering, 2019.

[29] Judea Pearl. Causal inference. Causality: Objectives and Assessment, pages 39–58, 2010.

[30] Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. In International Conference on Learning Representations, 2021.

[31] Florin Ratajczak, Mitchell Joblin, Martin Ringsquandl, and Marcel Hildebrandt. Task-driven knowledge graph filtering improves prioritizing drugs for repurposing. BMC bioinformatics, 23(1):1–19, 2022.

[32] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. ACM Transactions on Knowledge Discovery from Data (TKDD), 15(2):1–49, 2021.

[33] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You can teach an old dog new tricks! on training knowledge graph embeddings. 2020.

[34] 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.

[35] Babak Salimi, Harsh Parikh, Moe Kayali, Lise Getoor, Sudeepa Roy, and Dan Suciu. Causal relational learning. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 2020.

[36] Arjun Sondhi and Ali Shojaie. The reduced pc-algorithm: Improved causal structure learning in large random networks. J. Mach. Learn. Res., 20(164):1–31, 2019.

[37] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. Causation, prediction, and search. MIT press, 2000.

[38] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Yiqi Wang, Jiliang Tang, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. Investigating and mitigating degree-related biases in graph convoltuional networks. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pages 1435–1444, 2020.

[39] Sudhanshu Tiwari, Iti Bansal, and Carlos R Rivero. Revisiting the evaluation protocol of knowledge graph completion methods for link prediction. In Proceedings of the Web Conference 2021, pages 809–820, 2021.

[40] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Multi-task feature learning for knowledge graph enhanced recommendation. In The World Wide Web Conference, 2019.

[41] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019.

[42] Zihao Wang, Hang Yin, and Yangqiu Song. Benchmarking the combinatorial generalizability of complex query answering on knowledge graphs. In NeurIPS Datasets and Benchmarks Track, 2021.

[43] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised graph learning for recommendation. In Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, pages 726–735, 2021.

[44] 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.

[45] Hongming Zhang, Xin Liu, Haojie Pan, Yangqiu Song, and Cane Wing-Ki Leung. Aser: A large-scale eventuality knowledge graph. In Proceedings of the web conference 2020, pages 201–211, 2020.

[46] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In Advances in Neural Information Processing Systems, 2018.

[47] Xun Zheng, Chen Dan, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. Learning sparse nonparametric DAGs. In International Conference on Artificial Intelligence and Statistics, 2020.

It's FREE to join!

TCDE

tab.computer.org/tcde/

The Technical Committee on Data Engineering (TCDE) of the IEEE Computer Society is concerned with the role of data in the design, development, management and utilization of information systems.

- Data Management Systems and Modern Hardware/Software Platforms
- Data Models, Data Integration, Semantics and Data Quality
- Spatial, Temporal, Graph, Scientific, Statistical and Multimedia Databases
- Data Mining, Data Warehousing, and OLAP
- Big Data, Streams and Clouds
- Information Management, Distribution, Mobility, and the WWW
- Data Security, Privacy and Trust
- Performance, Experiments, and Analysis of Data Systems

The TCDE sponsors the International Conference on Data Engineering (ICDE). It publishes a quarterly newsletter, the Data Engineering Bulletin. If you are a member of the IEEE Computer Society, you may join the TCDE and receive copies of the Data Engineering Bulletin without cost. There are approximately 1000 members of the TCDE.

Join TCDE via Online or Fax

ONLINE: Follow the instructions on this page:

www.computer.org/portal/web/tandc/joinatc

FAX: Complete your details and fax this form to **+61-7-3365 3248**

Name _____

IEEE Member # _____

Mailing Address _____

_____

Country _____

Email _____

Phone _____

TCDE Mailing List

TCDE will occasionally email announcements, and other opportunities available for members. This mailing list will be used only for this purpose.

Membership Questions?

Xiaoyong Du
Key Laboratory of Data Engineering and Knowledge Engineering
Renmin University of China
Beijing 100872, China
duyong@ruc.edu.cn

TCDE Chair

Xiaofang Zhou
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD 4072, Australia
zxf@uq.edu.au

IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1314

Non-profit Org.
U.S. Postage
PAID
Los Alamitos, CA
Permit 1398