Bulletin of the Technical Committee on

Data Engineering

December 2022 Vol. 45 No. 4

IEEE Computer Society

Letters

Letter from the Editor-in-Chief . . . . . *Haixun Wang* 1
Letter from the Special Issue Editor . . . . . *Yangqiu Song* 2

Special Issue on Learning and Reasoning on Knowledge Graphs and Applications

Logical Queries on Knowledge Graphs: Emerging Interface of Incomplete Relational Data . . . . . *Zihao Wang, Hang Yin, and Yangqiu Song* 3
Knowledge Graph Comparative Reasoning for Fact Checking: Problem Definition and Algorithms . . . . . *Lihui Liu*, Ruining Zhao*, Boxin Du, Yi Ren Fung, Heng Ji, Jiejun Xu, and Hanghang Tong* 19
Harnessing Knowledge and Reasoning for Human-Like Natural Language Generation: A Brief Review . . . . . *Jiangjie Chen and Yanghua Xiao* 39
College-Related Question Answering based on Knowledge Graph . . . . . *Cheng Peng, Hao Jiang, Junnan Dong, and Xiao Huang* 60
Implicit Sentiment Analysis of Chinese Texts based on Contextual Information and Knowledge Enhancement . . . . . *Zhenghui Cao, Siqi Wang, Haofen Wang, and Wenqiang Zhang* 72
Hypergraph Clustering Network for Interaction Data . . . . . *Tianchi Yang, Luhao Zhang, Cheng Yang, Chuan Shi, Maodi Hu, Tao Li, and Dong Wang* 88
Distilling Causal Metaknowledge from Knowledge Graphs . . . . . *Yuan Meng, Yancheng Dong, Shixuan Liu, Chaohao Yuan, Yue He, Jian Pei, and Peng Cui* 102

Conference and Journal Notices

TCDE Membership Form 122

Letter from the Editor-in-Chief

The last time the Data Engineering Bulletin published a special issue on the topic of Knowledge Bases was more than six years ago. In the September 2016 issue, we asked leading researchers in this field for their thoughts on the present and future of Knowledge Bases.

At the time, there were two major focuses in the Knowledge Base field: data harvesting and heterogeneous data management. The primary concern for data harvesting was how to automatically build large, high-quality knowledge bases from Internet sources. The most difficult challenge in heterogeneous data management was determining the best way to model the data in a knowledge graph that allows applications to easily access knowledge.

It is clear that these two challenges are still pressing today. Building knowledge graphs remains a task with a low ROI because the universe of knowledge is too vast, and many methods are as impractical as boiling the ocean. Even though information extraction techniques have matured with the progress of machine learning in text processing, we still struggle to capture the tail facts, or facts mentioned only once or twice in the corpus. Furthermore, the issue of heterogeneous data management continues to stymie applications' ability to leverage knowledge graphs.

Nonetheless, much progress has been made in this area since our September 2016 issue. There has been a strong pull from applications due to their increasing needs for information and knowledge, and simutaneously a strong push from technology, fueled by the great process in machine learning, particularly in natural language processing.

The Data Engineering Bulletin will devote several issues to review the field of Knowledge Bases. This latest issue, curated by Yangqiu Song, shares a glimpse of the knowledge base landscape today. We are witnessing a shift of focus and methodology, with representation, data quality, and reasoning taking precedence. The first paper in the issue, for example, focuses on data incompleteness, which is a critical problem for large-scale knowledge graphs. Given the limitations of knowledge harvesting, the approach of combining facts and ML-powered estimations for every information need is promising. The second paper addresses another recurring issue of knowledge graphs: data inconsistency. It allows users to compare different outcomes of reasoning as the result of the inconsistency.

Haixun Wang
Instacart

Letter from the Special Issue Editor

Knowledge graphs, which were originally branded and popularized by Google in 2012, have attracted a lot of attention in both academia and industry. A knowledge graph is generally a graph-structured knowledge base that can leverage many graph processing and storage tools to perform knowledge representation and reasoning. Recently, machine learning, especially deep learning, has been widely used in many problems, including knowledge graph representation and reasoning. Thus, after more than ten years the term knowledge graph has been used, there are many new developments that are based on machine learning. First, machine learning models, particularly, graph representation learning, can be used to learn the representations of knowledge graph entities and relations. With the new representations, many reasoning tasks such as complex knowledge graph query, rule induction, knowledge base completion, and knowledge base population can be built based on learning models. Machine learning models can generalize better on knowledge graphs, where open-world assumption usually applies. Second, many new knowledge graph construction methods are based on machine learning. Information extraction and the recent "language model as a knowledge base" are all much improved with the development of scalable deep learning. With strong machine learning models, much less annotated data are required to construct many domain-specific knowledge graphs. Thus, many new types of knowledge graphs are recently developed. In this issue, we included several papers on learning and reasoning on knowledge graphs and applications.

The first paper *Logical Queries on Knowledge Graphs: Emerging Interface of Incomplete Relational Data* and the second paper *Knowledge Graph Comparative Reasoning for Fact Checking: Problem Definition and Algorithms* discussed the recent development of machine learning-based reasoning on knowledge graphs and its generalization ability to tackle the open world assumption problem of existing knowledge graphs. The following paper *Harnessing Knowledge and Reasoning for Human-Like Natural Language Generation: A Brief Review* surveyed the idea "language model as a knowledge base" which leverages the power of natural language generation to acquire knowledge for knowledge graph construction. Then the following three papers *College-Related Question Answering based on Knowledge Graph*, *College-Related Question Answering based on Knowledge Graph*, and *Implicit Sentiment Analysis of Chinese Texts based on Contextual Information and Knowledge Enhancement* studied applications of knowledge representation learning, knowledge-based question answering, and knowledge enhanced natural language processing problems. Finally, the last paper *Distilling Causal Metaknowledge from Knowledge Graphs* discussed how to distill causal metaknowledge from existing knowledge graphs, which is a very interesting and promising new direction of knowledge graph-related research.

I would like to thank all the authors for their contributions to make this issue an interesting discussion about present and future research directions related to knowledge graph representation learning and reasoning.

Yangqiu Song
The Hong Kong University of Science and Technology

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, 56, 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 [49, 62, 46, 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 [56] 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

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

graph representation or knowledge base completion [25, 45]. 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.

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 [55], 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 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 [51], 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

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, 58]
FB15k [53]	14,951	1,345	592,213	-	[42, 43, 58]
DBPedia [5]	34,575	3	240,942	Hierarchical KG	[14]
WN18RR [37]	40,903	11	103,509	Hierarchical KG	[24]
NELL955 [59]	63,361	200	142,804	-	[42, 43, 23, 58]
DRKG [64]	97,238	107	5,874,271	-	[13]
FB400k [50, 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]

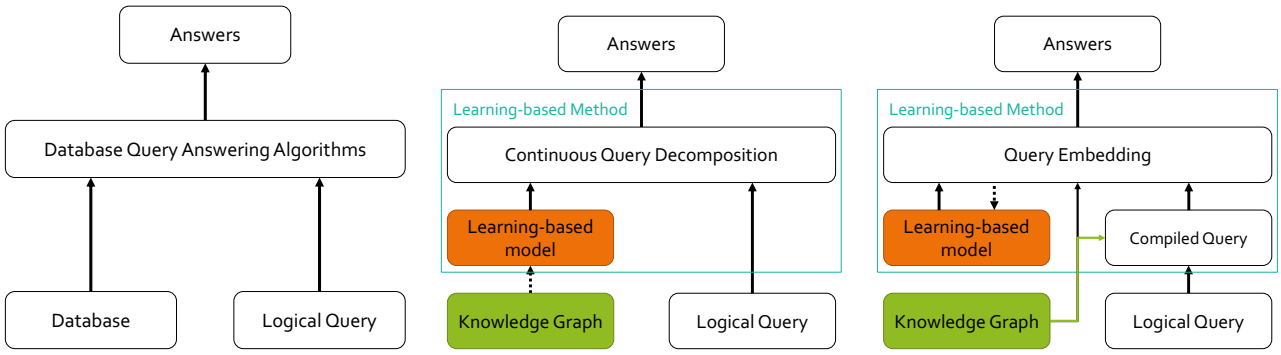
Table 2: Formal definitions of three typical query families. Compared to the first-order logic formula defined formally with Definition 6, three query families are defined using a subset of connectives or quantifiers (indicated by ✓).

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

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, 58] 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.



(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.

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].
- 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

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	EFO-1 †
	QA samples	Sequence + Model	EPFO †

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 [65, 25]. Specifically, CQD leverages the pretrained ComplEx embedding [54] 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 [45].

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 $\perp$ be the t-norm and t-conorm [20], we rewrite the EPFO query in Equation (1) into the optimization problem:

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

The objective in Problem (2) is continuous since p, $\top$, and $\perp$ 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

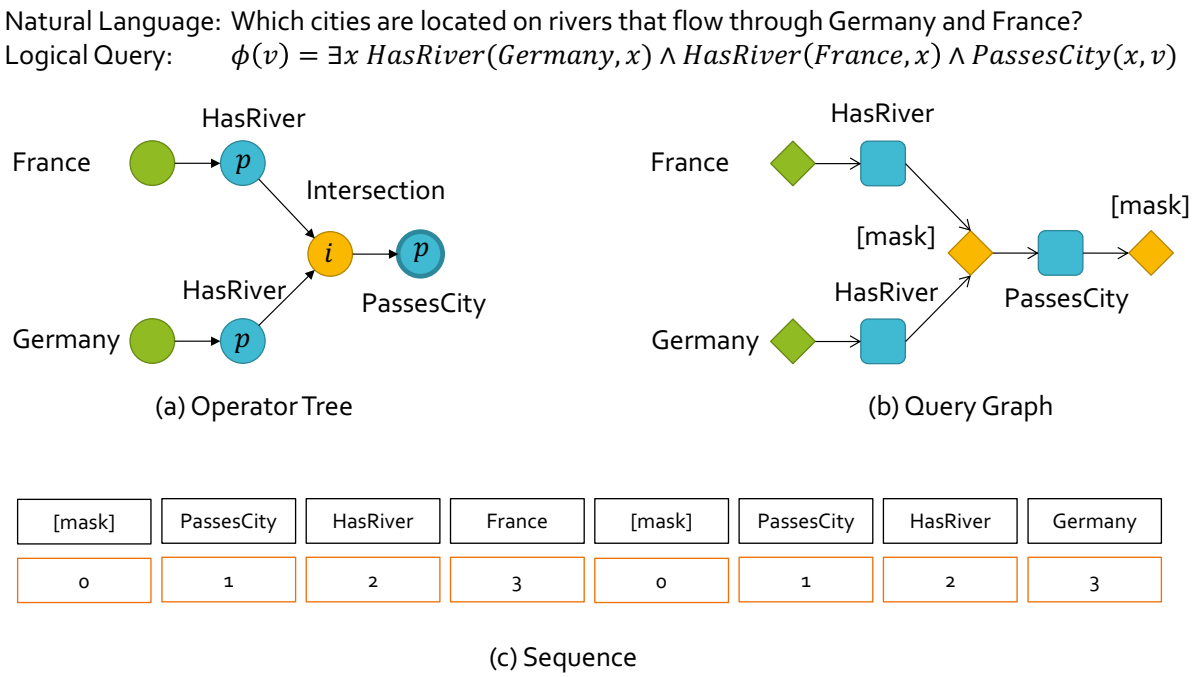


Figure 2: Three ways to represent logical query in the query embedding methods. The example is taken from [28]. We see the logical query is converted into three formats, including (a) operator tree [58], where each node is a set operator; (b) query graph [34], where the nodes are terms and predicates; (c) sequences [28], where a logical query is transformed as the concatenation of several multi-hop queries with a special positional encoding.

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.

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 [58], 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 [58]. 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 [67], 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 [63] to merge multiple regions into one. Notably, the design of ConE [67] 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 [61]. 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, 61]. Also, altering one parameter to its reciprocal is straightforward to represent the set complement for some family of distributions [43, 61]. 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, 61]. 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 [47] 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 [52] 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 [68] estimates the probabilities of all entities given

the set projection with graph neural network, NBFNet [69] over the observed knowledge graph. ENeSy [60] 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 in to graphs [16, 34, 57]. Query graph representation does not assume the acyclicity as operator tree representation. Therefore it represents a larger set of logical queries. However, existing query graphs are not related to the disjunction. The disjunction can be handled by converting existing queries to DNF queries, whose answer set is the combination of all conjunctive queries that can be addressed in the query graphs. More specifically, a conjunctive query is represented by two kinds of query graphs: (1) heterogenuous information networks (HIN) [48] where relations and entities are all nodes [34]; (2) constraint graphs where nodes are terms and edges are atomic formulas [4, 57]. This representation is connected to the constraint programming [44].

The learning-based models for query graphs are basically Graph Neural Networks (GNN). For constraint graph representation, MPQE [16] applies message passing networks on EPFO queries and LMPNN [57] employs the pretrained knowledge graph embeddings to answer queries with negation. 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 [66] 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, 46], 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 [55], 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.

7 Ackonwledgement

The authors of this paper were supported by the NSFC Fund (U20B2053) from the NSFC of China, the RIF (R6020-19 and R6021-20) and the GRF (16211520 and 16205322) from RGC of Hong Kong, the MHKJFS (MHP/001/19) from ITC of Hong Kong and the National Key R&D Program of China (2019YFE0198200) with special thanks to HKMAAC and CUSBLT, and the Jiangsu Province Science and Technology Collaboration Fund (BZ2021065). We also thank the support from NVIDIA AI Technology Center (NVAITC) and the UGC Research Matching Grants (RMGS20EG01-D, RMGS20CR11, RMGS20CR12, RMGS20EG19, RMGS20EG21, RMGS23CR05, RMGS23EG08).

References

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

[2] Dimitrios Alivanistos, Max Berrendorf, Michael Cochez, and Mikhail Galkin. Query Embedding on Hyper-Relational Knowledge Graphs. In International Conference on Learning Representations, March 2022.

[3] Alfonso Amayuelas, Shuai Zhang, Susie Xi Rao, and Ce 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] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex Query Answering with Neural Link Predictors. In International Conference on Learning Representations, 2021.

[5] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, The Semantic Web, Lecture Notes in Computer Science, pages 722–735, Berlin, Heidelberg, 2007. Springer.

[6] Jiaxin Bai, Zihao Wang, Hongming Zhang, and Yangqiu 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] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy 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] Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In Jason Tsong-Li 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] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

[11] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr, and Tom M. Mitchell. Toward an Architecture for Never-Ending Language Learning. In Maria Fox and David 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] Xuelu Chen, Ziniu Hu, and Yizhou 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] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan 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] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan 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] William Cohen, Fan Yang, and Kathryn Rivard Mazaitis. TensorLog: A Probabilistic Database Implemented Using Deep-Learning Infrastructure. Journal of Artificial Intelligence Research, 67:285–325, February 2020.

[16] Daniel Daza and Michael Cochez. Message Passing Query Embedding, June 2020.

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

[18] Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens 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, November 2020. Association for Computational Linguistics.

[19] Mikhail Galkin, Zhaocheng Zhu, Hongyu Ren, and Jian 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] Petr Hájek. Metamathematics of Fuzzy Logic, volume 4 of Trends in Logic. Springer Netherlands, Dordrecht, 1998.

[21] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding Logical Queries on Knowledge Graphs. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman 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] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure 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] Zhiwei Hu, Victor Gutierrez Basulto, Zhiliang Xiang, Xiaoli Li, Ru Li, and Jeff 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] Zijian Huang, Meng-Fen Chiang, and Wang-Chien 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, August 2022. Association for Computing Machinery.

[25] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems, 33(2):494–514, February 2022.

[26] Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. Complex Temporal Question Answering on Knowledge Graphs. In Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang 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, March 1977.

[28] Bhushan Kotnis, Carolin Lawrence, and Mathias 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] David M. Kroenke, David J. Auer, Scott L. Vandenberg, and Robert C. Yoder. Database Processing: Fundamentals, Design, and Implementation. Pearson, NY NY, 15th edition, 40th anniversary edition edition, 2018.

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

[31] Leonid Libkin and Cristina Sirangelo. Open and Closed World Assumptions in Data Exchange. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike 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] Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning. EMNLP/IJCNLP, 2019.

[33] Lihui Liu, Boxin Du, Heng Ji, ChengXiang Zhai, and Hanghang 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, August 2021. Association for Computing Machinery.

[34] Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie 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, August 2022. Association for Computing Machinery.

[35] Francois Luus, Prithviraj Sen, Pavan Kapanipathi, Ryan Riegel, Ndivhuwo Makondo, Thabang Lebese, and Alexander Gray. Logic Embeddings for Complex Query Answering. arXiv:2103.00418 [cs], February 2021.

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

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

[38] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia 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, April 2016. International World Wide Web Conferences Steering Committee.

[39] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander 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] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny 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] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Denny Zhou, Jure Leskovec, and Dale Schuurmans. SMORE: Knowledge Graph Completion and Multi-hop Reasoning in Massive Knowledge Graphs. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, pages 1472–1482, New York, NY, USA, August 2022. Association for Computing Machinery.

[42] Hongyu Ren, Weihua Hu, and Jure 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] Hongyu Ren and Jure Leskovec. Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien 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] Francesca Rossi, Peter Van Beek, and Toby Walsh. Handbook of constraint programming. Elsevier, 2006.

[45] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings. In International Conference on Learning Representations, March 2020.

[46] Apoorv Saxena, Soumen Chakrabarti, and Partha P. Talukdar. Question Answering Over Temporal Knowledge Graphs. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto 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.

[47] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, The Semantic Web, Lecture Notes in Computer Science, pages 593–607, Cham, 2018. Springer International Publishing.

[48] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. A survey of heterogeneous information network analysis. IEEE Transactions on Knowledge and Data Engineering, 29(1):17–37, 2016.

[49] Yueqing Sun, Qi Shi, Le Qi, and Yu Zhang. JointLK: Joint Reasoning with Language Models and Knowledge Graphs for Commonsense Question Answering. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza 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.

[50] Alon Talmor and Jonathan 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.

[51] Komal Teru, Etienne Denis, and William L Hamilton. Inductive Relation Prediction by Subgraph Reasoning. In Proceedings of the 37th International Conference on Machine Learning, pages 9448–9457. PMLR, November 2020.

[52] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. Advances in Neural Information Processing Systems, 34:24261–24272, 2021.

[53] Kristina Toutanova and Danqi 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.

[54] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. Complex Embeddings for Simple Link Prediction. In Proceedings of The 33rd International Conference on Machine Learning, pages 2071–2080. PMLR, June 2016.

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

[56] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. Communications of the ACM, 57(10):78–85, September 2014.

[57] Zihao Wang, Yangqiu Song, Ginny Y. Wong, and Simon See. Logical message passing networks with one-hop inference on atomic formulas. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali Rwanda, May 1-5, 2023. OpenReview.net, 2023.

[58] Zihao Wang, Hang Yin, and Yangqiu Song. Benchmarking the Combinatorial Generalizability of Complex Query Answering on Knowledge Graphs. In Joaquin Vanschoren and Sai-Kit Yeung, editors, Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, Virtual, 2021.

[59] Wenhan Xiong, Thien Hoang, and William Yang 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.

[60] Zezhong Xu, Wen Zhang, Peng Ye, Hui Chen, and Huajun Chen. Neural-Symbolic Entangled Framework for Complex Query Answering. In Advances in Neural Information Processing Systems, October 2022.

[61] Dong Yang, Peijun Qing, Yang Li, Haonan Lu, and Xiaodong Lin. GammaE: Gamma Embeddings for Logical Queries on Knowledge Graphs, October 2022.

[62] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao 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.

[63] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep Sets. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[64] Rui Zhang, Dimitar Hristovski, Dalton Schutte, Andrej Kastrin, Marcelo Fiszman, and Halil Kilicoglu. Drug repurposing for COVID-19 via knowledge graph completion. Journal of Biomedical Informatics, 115:103696, March 2021.

[65] Wen Zhang, Jiaoyan Chen, Juan Li, Zezhong Xu, Jeff Z. Pan, and Huajun Chen. Knowledge Graph Reasoning with Logics and Embeddings: Survey and Perspective. arXiv:2202.07412 [cs], February 2022.

[66] Yao Zhang, Peiyao Li, Hongru Liang, Adam Jatowt, and Zhenglu Yang. Fact-Tree Reasoning for N-ary Question Answering over Knowledge Graphs. In Smaranda Muresan, Preslav Nakov, and Aline 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.

[67] Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng 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.

[68] Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. Neural-Symbolic Models for Logical Queries on Knowledge Graphs, May 2022.

[69] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian 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.

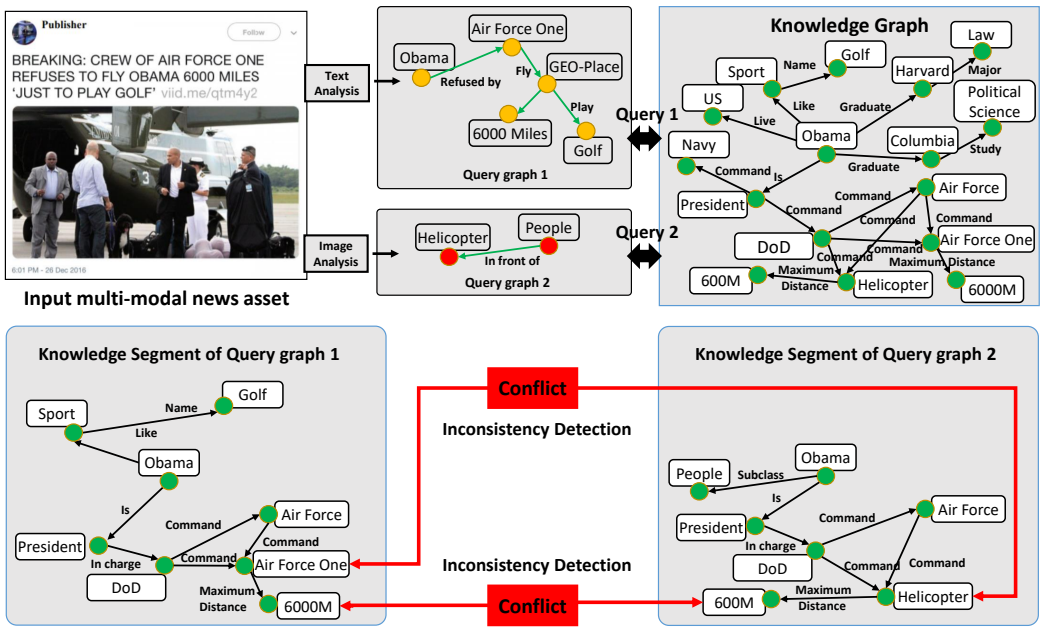


Figure 1: An illustrative example of using comparative reasoning for semantic inconsistency detection. Source of the image at the top-left: [4]. The example is borrowed from [18].

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 $i \neq \texttt{graduate}$, $U(\texttt{major}, i) = U(\texttt{study}, i) = 0$. Since $\text{E}(\texttt{graduate}) = 0$, we have $w(\texttt{graduate}) = 2\sigma(\infty) - 1 = 1$. We have TF(`major`, `graduate`) = TF(`study`, `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.

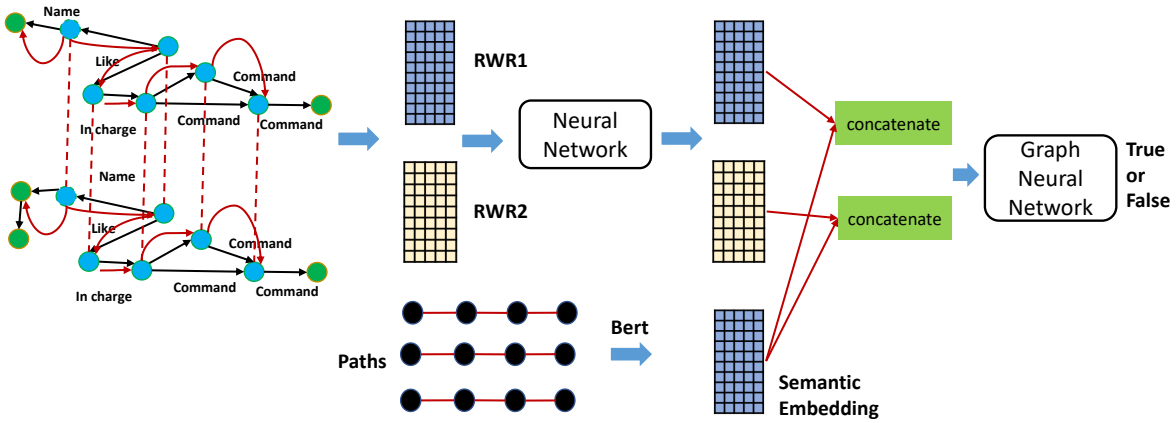


Figure 2: Neural network based pairwise comparative reasoning framework.

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 $<$ $\texttt{o}_1$, `isTypeOf`, $\texttt{o}_2$ $>$ and $<$ $\texttt{o}_2$, `isTypeOf`, $\texttt{o}_1$ $>$. 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

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

(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.

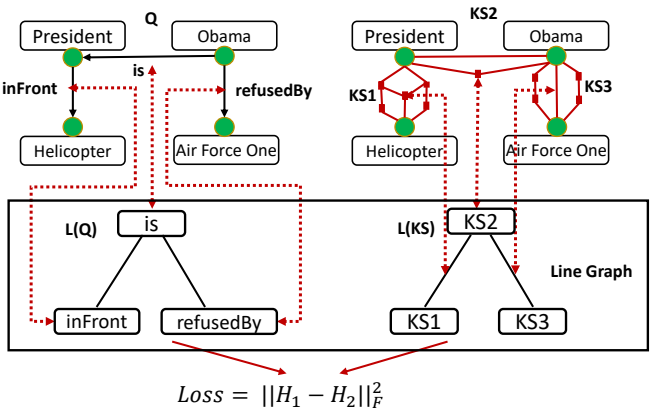


Figure 3: Collective comparative reasoning workflow.

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 $\pm$ 0.424	0.608 $\pm$ 0.302	0.736 $\pm$ 0.221	0.864 $\pm$ 0.105	0.594 $\pm$ 0.333	**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

Model \ Dataset	# 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 ± 0.313	0.658 ± 0.265	0.783 ± 0.155	**0.858** ± 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.

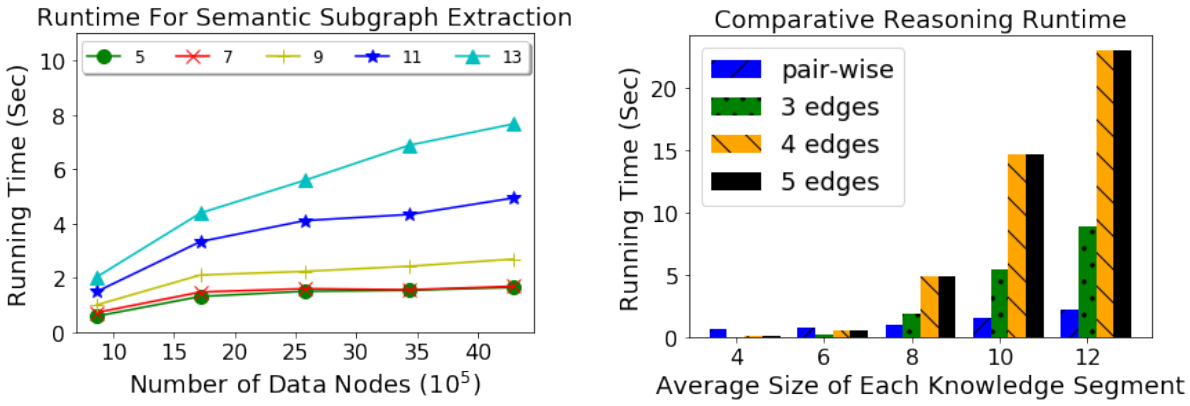
Dataset	# of queries	TransE	Jaccard	KL	KGMiner	Kompare
Positive	36	0.667	0.611	**1.000**	0.694	**1.000**
Negative	36	0.528	0.361	0.722	0.553	**0.863**
Average accuracy	-	0.598 ± 0.071	0.486 ± 0.126	0.861 ± 0.138	0.623 ± 0.071	**0.932** $\pm$ **0.063**

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.

Harnessing Knowledge and Reasoning for Human-Like Natural Language Generation: A Brief Review

Jiangjie Chen
Fudan University
jjchen19@fudan.edu.cn

Yanghua Xiao
Fudan University
shawyh@fudan.edu.cn

Abstract

The rapid development and application of natural language generation (NLG) techniques has revolutionized the field of automatic text production. However, these techniques are still limited in their ability to produce human-like text that is truly reasonable and informative. In this paper, we explore the importance of NLG being guided by knowledge, in order to convey human-like reasoning through language generation. We propose ten goals for intelligent NLG systems to pursue, and briefly review the achievement of NLG techniques guided by knowledge and reasoning. We also conclude by envisioning future directions and challenges in the pursuit of these goals.

1 Introduction

Language, as the vehicle of thought [128], is one of the most fundamental means for humans to reason and communicate with each other. Hence, the technology of natural language generation (NLG) has always been one of the main focuses throughout the history of AI research [126]. In the age of modern deep learning, NLG techniques have been widely deployed in real-life applications, including machine translation [130], news reporter [134], dialogue systems [115], automatic report generation [40], etc. Beyond that, NLG models also serve as a rather universal workhorse in other types of NLP tasks, such as structured prediction [85].

The achievement so far for NLG research has enabled the wide application of NLG techniques, but the dangers beneath them are still far from being resolved. The foundation of NLG models has shifted over the years, from rule-based models [62, 81] to statistical models [27], and now at pre-trained language models (PLMs) [92, 34, 93, 64, 11]. However, rule-based models are too rigid to generate natural language. Statistical models based on neural networks and training datasets suffer from limitations such as reporting bias, exposure bias, generalization issue, etc. Recent years, PLMs have shown their excellent capabilities of understanding and generating natural language. However, the research on PLMs does not necessarily solve the fundamental problems NLG, as the alleviation of these problems comes from exposing the models on much more data with self-supervised training. Moreover, multiple problems still occur, including model explainability [72], hallucination [95, 132], ethical risks [117, 148], logical inconsistency [60, 36], etc.

Towards these challenges, the research community has formulated an important perspective with symbolic knowledge, covering rules [55], commonsense knowledge [109], world knowledge [2, 120], etc. Since symbolic

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

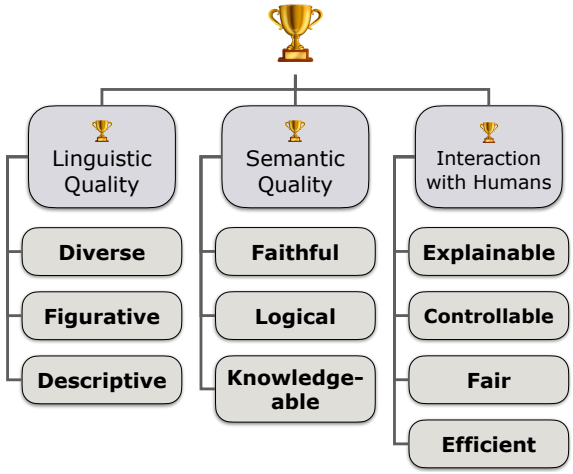


Figure 1: The taxonomy of the goals in NLG.

knowledge can be used to describe complicated concepts and their connections, it would be easier for the models to comprehend the textual world. Therefore, symbolic knowledge brings significant opportunities towards NLG models in these respects: 1) symbolic knowledge provides rich background materials for NLG models to generate from; 2) symbolic knowledge serves as regularization to NLG models for expressing constraints and desired properties related to a given task; and 3) the formal rules and structures that symbolic knowledge established can be used to improve the reasoning ability of NLG models.

In this paper, we stress the importance and necessity of NLG techniques to be guided by knowledge, so that human-like reasoning can be conveyed through language generation. To this end, we propose ten goals for intelligent NLG systems to pursue (§ 2), Next, we briefly review the achievement of NLG techniques with knowledge (§ 3), picturing the mutual enhancement of guiding NLG with knowledge and acquiring knowledge with NLG. Then, we summarize how to make rational usage of knowledge to approach human-level reasoning with NLG techniques (§ 4), showing reasoning-guided NLG and how NLG can be used to chain up reasoning. Finally, we conclude this paper and envision future directions and challenges in the pursuit of these goals (§ 5).

2 Holy Grails in NLG

To begin with, we list ten most-desired goals (with room for more) for machine-generated text, in order to build an AI system that can freely interact and communicate with humans with language. We categorize these goals into three classes based on the linguistic quality of generated text, conveying semantic information with generated text and the interaction with humans, as shown in Figure 1. These goals are still quite challenging for modern NLG methods. While the research about most of them is still in its infancy, some of these goals have been moderately explored by the community. Since symbolic knowledge offers control of the process and outcome of generated text, many of research endeavors prove that symbolic knowledge plays an important role in steering NLG techniques towards these goals.

Diverse Given a source input, an ideal NLG model needs to be able to generate multiple and diverse outputs, where each of them needs to be equally valid. For example Diversified NLG is a practical goal, especially in the context of real-life applications such as dialogue generation [129], machine translation [101], headline or query generation in e-commerce [102] and text paraphrasing [103]. Advanced NLG models should be able to generate diverse but informative text from the large sentence space, especially involving diverse but relevant background knowledge in the context.

Figurative The generative text should be figurative to build up emotional significance to the readers. Figurative text typically includes idioms, sarcasm, simile, metaphor, etc. All of them are great ingredients in interesting and creative writing tasks such as story generation or narratives. For example, the sentence that "he is drowning in a sea of grief" expresses strong negative emotions to the reader, where *grief*, with the metaphor of a *sea*, vividly overwhelms *him*. To master the ability to write figurative text is non-trivial, and recent studies show that even strong modern language models still struggle at this objective [47, 70, 22]. Nevertheless, research shows that such a problem can be alleviated with knowledge-enhanced models, which enrich the context and constituents of figurative text with acquired knowledge [18].

Descriptive Advanced NLG techniques need to be descriptive, that is, vivid and colorful as if something is being experienced by the readers. Descriptive texts are rather common in books and novels, fascinated by image-like texts. For example, to describe the sunset: "*the sunset filled the entire sky with the deep color of rubies, setting the clouds ablaze.*"[1] It is worth noting that, unlike previous desired properties that interact mostly with textual data, a system needs to integrate multi-modal knowledge and reasoning (e.g., image, video, etc.) to be descriptive with visual-specific features [138, 105, 106]. However, the exact definition of descriptiveness in the context of the machine-generated text as well as its automatic evaluation are still great challenges.

Faithful The generated text should be faithful to the input so that it correctly conveys and extends the input information without semantic violation. Otherwise, the credibility of an NLG system could be undermined when hallucinated texts are generated, which could be dangerous sometimes. There is a growing interest in enhancing and evaluating the (intrinsic or extrinsic) faithfulness of generated text in text summarization [17, 80, 63], dialogue systems [49], text simplification [33], etc. However, hallucinated texts are not always non-factual, because they may be faithful towards world knowledge [14]. Therefore, an NLG system should be faithful and can be verified by world knowledge [116, 20], except for applications such as fictional story generation.

Logical Logical reasoning is an essential part of human thinking and language; therefore, the generated text needs to be logically consistent and self-contained. Different from faithfulness which focuses on information consistency, logical consistency poses a challenge over the discourse of produced language of an NLG system [5, 25, 104, 107, 87]. However, the training objectives of current prevailing language modeling (e.g., masked language modeling and causal language modeling) prioritize recovering given text, which does not guarantee the logical reasoning ability to be effectively captured by models.

Knowledgeable An NLG model needs to be knowledgeable to generate text that is rich in knowledge [139]. When engaging in a conversation, current NLG models are known to be dependent on plain but safe responses. A knowledgeable NLG system, in contrast, should be able to actively initiate responses grounded with the background knowledge that is either retrieved [65] or inherent [75] in the system itself. Also, it is worth exploring whether current NLG models are using knowledge of their own or just relying on statistical patterns learned from (pre-)training [13].

Explainable An NLG system should be explainable for the trust of human users. Since language is the most natural tool for communication, the reasoning and decisions of a model should be explanatory through generated language [96]. Existing work usually focuses on generating natural language explanations [127] as a task, while how to develop universal explanation generators [137] is still challenging. More importantly, unlike natural language understanding models (NLU) [97, 100, 111], the explainability of NLG systems is severely underexplored.

[1] https://rescuewriting.org/featured/writing-descriptive-text/.

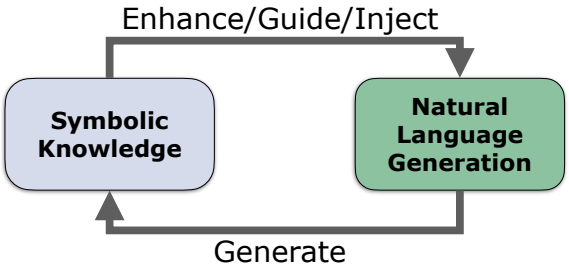


Figure 2: Relations between knowledge and NLG, where knowledge can enhance, guide and be injected into NLG systems, and NLG systems can be utilized as tools to generate new knowledge.

Controllable Controllable NLG aims to generate with desired attributes by users, which greatly broadens its applications. Current commonly used attributes focus on discriminative attributes such as sentiment and style, lexical constraints, and various properties of language such as lengths or complexity [50, 141, 37]. Moreover, there are still many interesting open questions for building an ideal NLG system, for example, more diverse applications of controlling factors, satisfying multiple types of constrained attributes (perhaps a mixture of them), and being controlled in a few-shot or even zero-shot manner. It is worth noting that recent studies in large language models [11] shed some light on these research problems.

Fair The generated text should be fair and contain no bias whatsoever. Methods to mitigate biases have been proposed w.r.t. gender, race, nationality, age, disability, religion, etc. [71, 131, 45]. Since text generation models have been widely applied in the age of the Internet, their outstanding performance could blind service providers so as to omit the negative social impacts that an unfair NLG system causes. However, since modern NLG systems are usually pre-trained on colossal unannotated corpus [92, 93, 64, 11], the biases within them are still rather difficult to eliminate.

Efficient An NLG system should be efficient to be deployed on high-demanding scenarios such as mobile devices or online industrial applications. Recent work on non-autoregressive generation (NAG) [41, 133], which generates tokens in parallel, has shown promising potential in efficiency. This enables NAG to achieve over $10\times$ speedup over the autoregressive counterpart (AG) that generates in a left-to-right manner [27]. However, NAG is still haunted by the multi-modality problem [145], its generality to tasks other than machine translation [89], and interaction with external knowledge [140].

3 Knowledge in Language Generation

In this section, we briefly review some representative work in knowledge-guided NLG algorithms and knowledge acquisition with NLG models, where NLG systems also show unique advantages in the latter. Figure 2 shows the relation between knowledge-guided NLG (§ 3.1) and knowledge acquisition with NLG (§ 3.2), where the two are mutually beneficial to each other.

3.1 Knowledge-guided NLG: Tricks of the Trade

NLG systems can be guided by multiple sources and types of knowledge to be logical, diverse, faithful, controllable, etc. In the following text, we list the knowledge sources that are commonly used, and showcase the common tricks of current knowledge-guided NLG algorithms, presenting a brief but as comprehensive review as possible.

Knowledge Sources According to Yu et al. [139], knowledge sources used in NLG systems can be categorized into internal and external knowledge, where: 1) internal knowledge is created within the given text, e.g., keywords, topics, linguistic features, etc., and 2) external knowledge exists outside the input of the NLG system, e.g., knowledge bases/graphs, grounded unstructured text. We add that knowledge mined from the corpus is also important to guide NLG systems, such as concepts, rules, and patterns [21]. Notably, NLG systems form an interesting dual learning loop where knowledge can be acquired by NLG systems (§ 3.2) and used in NLG systems [110, 16]. Next, we will detail some common practices of NLG models that involve knowledge.

Knowledge Incorporation in Model Architectures Modifying model architectures to incorporate knowledge is one of the most commonly used approaches in knowledge-guided NLG [136, 139]. Typical methods include using attention mechanism [3] (and its variants, e.g., copying mechanism, [42]) to attend additional knowledge sources, encoding knowledge with specific encoders (e.g., graph neural networks, [61]), adding specific layers in the neural network for knowledge storage (e.g., adapter layers, [44]), etc. In this line of work, a model trains to utilize the encoded knowledge for better NLG. However, such methods usually require the design of an ad hoc model architecture, which does not generalize well to new tasks, especially in the age of large PLMs [11].

Learning with Knowledge The most common method of knowledge-guided NLG is through supervised learning, where the guidance of knowledge can be well integrated into generation [139]. A simple way to do this is to append the acquired knowledge to the input as an additional context. This has been shown to be effective, especially when knowledge acquisition (e.g., retrieval) is jointly trained with generation [46, 65]. Another way is to guide learning objectives with knowledge. Studies in this direction are carried out by designing knowledge into one of the training targets [16] or pre-training [43], or by introducing additional knowledge-related objective functions to regularize training [51]. Also, reward design in a reinforcement learning (RL) framework is a flexible way of integrating knowledge into training objectives for NLG models. For an example of improving the consistency between source and target, [67] injects the entailment knowledge into NLG with an entailment model, and [53] designs a semantic cloze reward for faithful summarization.

Knowledge-Constrained Decoding Finally, we would like to emphasize knowledge incorporation during decoding. Knowledge injection through model architecture modification and new learning objectives, despite being effective, requires (re-)training. This usually causes inconvenience for deployment, especially with large PLMs. For this reason, constraining text decoding in inference time with knowledge has become a promising research topic [141, 37]. In this line of work, we briefly introduce the three most exemplary knowledge constraint types in NLG systems: 1) lexical knowledge constraints, such as keywords that must appear in the text, 2) discriminative constraints, such as the desired attributes of the generated text, and 3) structured knowledge constraints, such as the world knowledge that the generated text must be consistent with.

Imposing *lexical constraints* is challenging for NLG systems because the prevailing ones usually generate in an autoregressive manner, which is difficult to know when and where to keep the constraints. Popular solutions to this problem mainly focus on adding constraints during beam search [48, 88, 76] at the cost of computational complexity. Recent work [112, 135] also explores iterative lexically constrained decoding for non-autoregressive models, achieving significant speedups.

Discriminative constraints usually involve prior attribute models to control the generated text in terms of attributes such as topics (sport, finance, medical), sentiments (positive, negative), etc. Exemplary work is PPLM [30], which controls the attribute of generated text without changing the parameters of the backbone PLM (e.g., GPT-2, [92]). During sampling, PPLMs back-propagates the gradient from the attribute model to the hidden states of the PLM so that the generation can be steered by the attribute model. Under such a framework, it is easy to guide generation with prior knowledge. It is worth noting that work on text sampling is also a great framework to effectively incorporate the above-mentioned two types of constraints. In this framework, the desired

properties are designed as objective functions, guiding the sampling process, which is usually instantiated with Markov chain Monte Carlo methods [82, 143, 91]. Similar to PPLM, such methods ensure the generated text is constrained to the desired objectives without training.

Recent studies have also started to pay attention to the hallucination problem of NLG systems (represented by PLMs) from the perspective of constrained decoding with *structured knowledge constraints*, such as knowledge graphs. For example, [73] retrieves from knowledge graphs related to neighboring entities within source input. They are used to modify and constrain the distributions over the vocabulary during sampling, where tokens within knowledge graphs are rewarded. In this way, the generated text would be rich in semantic knowledge and faithful to world knowledge, making it an interesting and effective solution to the hallucination problem that is common in NLG.

3.2 Knowledge Acquisition with NLG

Knowledge acquisition based on NLG systems aims to generate knowledge from the input text. Compared with other paradigms of knowledge acquisition, automatic and generalizability to unseen knowledge are one of the most desired features of generative methods. In the following text, we first discuss current knowledge acquisition paradigms and show the advantages of generative methods in certain scenarios.

Paradigms of Knowledge Acquisition The knowledge acquisition methods can be divided into three categories: 1) crowd-sourcing, 2) extractive and 3) generative methods. The *crowd-sourcing* methods [83] invite some experts to label the knowledge in the corpus, which can acquire accurate knowledge but only have a small size due to costly annotation. The *extractive* methods [147] adopt language models to extract the knowledge explicitly mentioned in the input text automatically but ignore some symbolic and neural commonsense knowledge. Compared with them, generative knowledge acquisition methods enjoy the advantages of the ability to generate explicit and implicit knowledge, which we will soon discuss.

Meta Knowledge Generation Meta knowledge is knowledge about knowledge [31]. As a fundamental conceptual instrument in knowledge-based domains, meta-knowledge can greatly improve the performance of downstream tasks, such as text classification [23], question answering [38] and story generation [19]. However, meta-knowledge is hardly explicitly mentioned in the corpus, and thus it is difficult to directly extract meta-knowledge from the text. Alternatively, since NLG systems can generate information never mentioned in the text and has a strong generalization ability of unseen knowledge, recent work proposes to adopt generative methods to acquire meta knowledge, such as concepts [21, 66] and rules [24].

Generative Knowledge Retrieval Information retrieval aims to retrieve meaningful information from large Knowledge Bases (KB) given a textual input. Compared to extractive methods, generative information retrieval [15, 32, 98] can directly capture the relation between context and target information and reduce the memory footprint without negative data down-sampling.

Generative Knowledge Extraction Information extraction aims to obtain information from semi-structured and unstructured text, which suffers heterogeneous structures and domain-specific schemas [79]. The generative information extraction can end-to-end generate targeted structures directly. For example, [54] adopts autoregressive models to achieve the extraction of end-to-end relations. [78] proposes a sequence-to-structure generation method to directly extract events from the text. [52] formulates entity-based extraction as a template generation task to allow the generative framework to effectively capture cross-entity dependencies.

Knowledge Generation NLG models can also be used for the completion of knowledge bases, including commonsense knowledge graphs and rules. Instead of extracting semi-structured and unstructured text into knowledge, some work [9, 55] feeds large-scale language models with a massive corpus to obtain knowledge models, which can adapt their learned representations to knowledge generation and automatically construct KBs. Furthermore, the parameters of PLMs are shown to store vast amounts of linguistic knowledge [114]. A line of work further regards PLMs as knowledge bases and distills semantic knowledge from these models [86, 1].

4 Reasoning in Language Generation

We echo the argument that good reasoning should be right for the right reasons. Therefore, it is also crucial for an NLG model to make *rational* usage of knowledge to approach human-like reasoning skills. This section signifies the importance of reasoning in NLG, where we sketch two lines of research: 1) reasoning-guided NLG methods (§ 4.1) and 2) NLG for the purpose of reasoning (§ 4.2).

4.1 Reasoning-guided NLG

In contrast to general knowledge-guided NLG, reasoning-guided NLG systems make more rational and explainable usage of the knowledge (in wide forms). Since reasoning is highly correlated with knowledge (§ 3), we will discuss reasoning-guided NLG by highlighting the topics of graph reasoning and generative reasoning tasks in the following paragraphs.

Graph Reasoning Graph reasoning is one of the most commonly used techniques to guide NLP systems towards more multihop, controllable, and explainable reasoning. Therefore, we extend what has been discussed in § 3.1 in this paragraph for introducing graph reasoning-guided NLG. Graph reasoning implementations are usually built on retrieved subgraphs of external knowledge graphs [74] or internal graphs parsed from the input [121]. Most of them adopt graph embeddings [8] and graph neural networks [61, 119] to propagate information throughout the graph. These properties of graphs enable the multi-hop reasoning ability of NLG models for long-range text [57] and generating emerging concepts or topics guided by the graph [122, 142]. Due to the symbolic structure of graphs, heuristics and prior knowledge can be easily incorporated into graph construction, e.g., building graphs with various parsing tools [39] such as semantic role labeling or dependency parsing or guiding graph propagation with more information, such as popularity knowledge [102]. Also, such methods are explainable and easy to debug, since the weights on the graph nodes and edges greatly facilitate post hoc manual examination.

Reasoning Tasks One of the most direct ways to enable NLG models to reason is to design various reasoning tasks. During the solving of these tasks, researchers can develop and test their models w.r.t. corresponding reasoning skills, which makes it an important direction to guide AI models. Over the years, the community has accumulated many datasets of tasks that test various facets of machine reasoning in the form of text generation. Most of them are built from the point of view of human cognition, including tasks about logical reasoning [29], abductive reasoning [6], counterfactual reasoning [90], generative commonsense reasoning [69], social reasoning [99], physical reasoning [7], temporal reasoning [144], etc. [2] Not limited to these tasks, datasets on explanation generation [127] are also good sources to evaluate and improve the reasoning ability of NLG systems, including the explanations for natural language inference [12] commonsense reasoning [94], analogical reasoning [22], causal reasoning [35], multimodal reasoning [68], etc.

[2] Note that some of these reasoning tasks [99, 7, 144] take the form of question answering but can be solved in a generative manner [58].

4.2 Reasoning by NLG

Human language is a good vehicle for reasoning. Thus, the generation of language naturally resembles the way humans think and reason. With the success of PLMs, there is a growing interest in the AI community to use NLG models to generate a chain of reasoning for problem-solving.

Generative Reasoning Generative reasoning aims to generate intermediate reasons with NLG models for better problem solving. Such intermediate reasons take many forms, including deduction and abduction reasons [113], explanations [56], or decomposed subtasks of a complex one [59]. Some work [108, 4] even show that expanding the context of the input by generating more information would also help solve reasoning tasks. Moreover, [27] finds that transformer-based PLMs are effective soft reasoners on a toy deduction dataset, which consists of collections of text verbalized from artificial if-then rules and facts. Other studies [10, 5] corroborate this discovery and show that training generative models with artificial textual data that verbalize rule-based reasoning helps downstream logical reasoning tasks. We remark that generative reasoning provides a new perspective of breaking the black-box prediction of neural networks, which demonstrates the potential of achieving reasoning with NLG systems.

Reasoning with Large Language Models Entering the era of large language models (LLMs) such as GPT-3 [11, 84] and PaLM [26], there is a recent growing interest in exploring few-/zero-shot reasoning skills of these LLMs Since fine-tuning such tremendous language models is hardly possible, current work adopts prompt-based in-context learning methods [11, 77] to achieve few-/zero-shot learning with LLMs, where instructions and examples are demonstrated within the input prompts.

Similar to the above discussion of generative reasoning, this line of work aims to guide the language models to explicitly generate the intermediate thinking steps (or reasons) during reasoning. A representative work among them is the Chain-of-Thought prompting [125], where the intermediate thinking process is verbalized and integrated into the demonstrations. In this way, complex reasoning can be decomposed into multiple steps reflected by language, which is analogous to how humans solve complex tasks. Such methods achieve much better performance on a variety of reasoning tasks compared with normal prompting and even surpass fine-tuned methods in some cases. Moreover, the prompting strategy can be further refined [123, 146, 28], leading to generally better results. LLMs prompted with and asked to generate step-by-step reasoning chains also exhibit certain quantitative reasoning abilities such as solving math word problems [125], where the LLMs are not specifically trained on such tasks. However, reasoning abilities for LLMs are shown to be emergent [125, 124], i.e., effective only for really large language models (over 100 billion parameters). How to enable smaller language models with few-shot reasoning skills is still an open question.

5 Conclusion

In this work, we envision the ten most desired goals of an intelligent natural language generation system. In pursuit of these goals, the guidance of knowledge and reasoning plays a significant role in modern NLG models. We have revisited the achievements with knowledge in NLG w.r.t. knowledge-guided NLG and generative knowledge acquisition. We particularly highlight knowledge-constrained decoding for its wide application potential, where knowledge constraints can be incorporated into NLG models (especially large-scale ones) in a plug-and-play manner. We have also discussed current work on reasoning in NLG, which essentially makes rational usage of knowledge and datasets for various reasoning tasks. Moreover, NLG can verbalize intermediate thinking processes to facilitate complex reasoning, enabling few-/zero-shot reasoning abilities for large language models. However, current research is still far from realizing these goals, which we outline for future research. We hope

that this survey report can provide newcomers with a good entry point into the exciting area of knowledge-guided and reasoning-intensive NLG.

References

[1] Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. A review on language models as knowledge bases. arXiv preprint arXiv:2204.06031, 2022.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference, ISWC'07/ASWC'07, page 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[4] Gregor Betz, Kyle Richardson, and Christian Voigt. Thinking aloud: Dynamic context generation improves zero-shot reasoning performance of gpt-2. arXiv preprint arXiv:2103.13033, 2021.

[5] Gregor Betz, Christian Voigt, and Kyle Richardson. Critical thinking for language models. In Proceedings of the 14th International Conference on Computational Semantics (IWCS), pages 63–75, Groningen, The Netherlands (online), June 2021. Association for Computational Linguistics.

[6] Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen tau Yih, and Yejin Choi. Abductive commonsense reasoning. In International Conference on Learning Representations, 2020.

[7] Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. Proceedings of the AAAI Conference on Artificial Intelligence, 34(05):7432–7439, Apr. 2020.

[8] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013.

[9] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. COMET: Commonsense transformers for automatic knowledge graph construction. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4762–4779, Florence, Italy, July 2019. Association for Computational Linguistics.

[10] Kaj Bostrom, Xinyu Zhao, Swarat Chaudhuri, and Greg Durrett. Flexible generation of natural language deductions. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 6266–6278, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark,

Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[12] Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: Natural language inference with natural language explanations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018.

[13] Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. Knowledgeable or educated guess? revisiting language models as knowledge bases. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1860–1874, Online, August 2021. Association for Computational Linguistics.

[14] Meng Cao, Yue Dong, and Jackie Cheung. Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3340–3354, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[15] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In International Conference on Learning Representations, 2021.

[16] Ruisheng Cao, Su Zhu, Chenyu Yang, Chen Liu, Rao Ma, Yanbin Zhao, Lu Chen, and Kai Yu. Unsupervised dual paraphrasing for two-stage semantic parsing. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6806–6817, Online, July 2020. Association for Computational Linguistics.

[17] Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. Faithful to the original: Fact aware neural abstractive summarization. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[18] Tuhin Chakrabarty, Yejin Choi, and Vered Shwartz. It's not rocket science: Interpreting figurative language in narratives. Transactions of the Association for Computational Linguistics, 10:589–606, 2022.

[19] Hong Chen, Yifei Huang, Hiroya Takamura, and Hideki Nakayama. Commonsense knowledge aware concept selection for diverse and informative visual storytelling. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 999–1008, 2021.

[20] Jiangjie Chen, Qiaoben Bao, Changzhi Sun, Xinbo Zhang, Jiaze Chen, Hao Zhou, Yanghua Xiao, and Lei Li. Loren: Logic-regularized reasoning for interpretable fact verification. Proceedings of the AAAI Conference on Artificial Intelligence, 36(10):10482–10491, Jun. 2022.

[21] Jiangjie Chen, Ao Wang, Haiyun Jiang, Suo Feng, Chenguang Li, and Yanghua Xiao. Ensuring readability and data-fidelity using head-modifier templates in deep type description generation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2036–2046, Florence, Italy, July 2019. Association for Computational Linguistics.

[22] Jiangjie Chen, Rui Xu, Ziquan Fu, Wei Shi, Zhongqiao Li, Xinbo Zhang, Changzhi Sun, Lei Li, Yanghua Xiao, and Hao Zhou. E-KAR: A benchmark for rationalizing natural language analogical reasoning. In Findings of the Association for Computational Linguistics: ACL 2022, pages 3941–3955, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[23] Jindong Chen, Yizhou Hu, Jingping Liu, Yanghua Xiao, and Haiyun Jiang. Deep short text classification with knowledge powered attention. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 6252–6259, 2019.

[24] Lihan Chen, Sihang Jiang, Jingping Liu, Chao Wang, Sheng Zhang, Chenhao Xie, Jiaqing Liang, Yanghua Xiao, and Rui Song. Rule mining over knowledge graphs via reinforcement learning. Know.-Based Syst., 242(C), apr 2022.

[25] Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. Logical natural language generation from open-domain tables. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7929–7942, Online, July 2020. Association for Computational Linguistics.

[26] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311, 2022.

[27] Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In Christian Bessiere, editor, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pages 3882–3890. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.

[28] Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. arXiv preprint arXiv:2205.09712, 2022.

[29] Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. Explaining answers with entailment trees. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7358–7370, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[30] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In International Conference on Learning Representations, 2020.

[31] Randall Davis and Bruce G. Buchanan. Meta-level knowledge: Overview and applications. In IJCAI, 1977.

[32] Nicola De Cao, Ledell Wu, Kashyap Popat, Mikel Artetxe, Naman Goyal, Mikhail Plekhanov, Luke Zettlemoyer, Nicola Cancedda, Sebastian Riedel, and Fabio Petroni. Multilingual Autoregressive Entity Linking. Transactions of the Association for Computational Linguistics, 10:274–290, 03 2022.

[33] Ashwin Devaraj, William Sheffield, Byron Wallace, and Junyi Jessy Li. Evaluating factuality in text simplification. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7331–7345, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[35] Li Du, Xiao Ding, Kai Xiong, Ting Liu, and Bing Qin. e-CARE: a new dataset for exploring explainable causal reasoning. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 432–446, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[36] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. Transactions of the Association for Computational Linguistics, 9:1012–1031, 2021.

[37] Cristina Garbacea and Qiaozhu Mei. Why is constrained neural language generation particularly challenging? arXiv preprint arXiv:2206.05395, 2022.

[38] François Gardères, Maryam Ziaeefard, Baptiste Abeloos, and Freddy Lecue. ConceptBert: Concept-aware representation for visual question answering. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 489–498, Online, November 2020. Association for Computational Linguistics.

[39] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In Proceedings of Workshop for NLP Open Source Software (NLP-OSS), pages 1–6, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[40] Dimitra Gkatzia, Oliver Lemon, and Verena Rieser. Data-to-text generation improves decision-making under uncertainty. IEEE Computational Intelligence Magazine, 12:10–17, 2017.

[41] Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In International Conference on Learning Representations, 2018.

[42] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1631–1640, Berlin, Germany, August 2016. Association for Computational Linguistics.

[43] Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. A knowledge-enhanced pretraining model for commonsense story generation. Transactions of the Association for Computational Linguistics, 8:93–108, 2020.

[44] Junliang Guo, Zhirui Zhang, Linli Xu, Hao-Ran Wei, Boxing Chen, and Enhong Chen. Incorporating bert into parallel sequence decoding with adapters. Advances in Neural Information Processing Systems, 33:10843–10854, 2020.

[45] Umang Gupta, Jwala Dhamala, Varun Kumar, Apurv Verma, Yada Pruksachatkun, Satyapriya Krishna, Rahul Gupta, Kai-Wei Chang, Greg Ver Steeg, and Aram Galstyan. Mitigating gender bias in distilled language models via counterfactual role reversal. In Findings of the Association for Computational Linguistics: ACL 2022, pages 658–678, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[46] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In Hal Daumé III and Aarti Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 3929–3938. PMLR, 13–18 Jul 2020.

[47] Qianyu He, Sijie Cheng, Zhixu Li, Rui Xie, and Yanghua Xiao. Can pre-trained language models interpret similes as smart as human? In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7875–7887, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[48] Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[49] Or Honovich, Leshem Choshen, Roee Aharoni, Ella Neeman, Idan Szpektor, and Omri Abend. q^2: Evaluating factual consistency in knowledge-grounded dialogues via question generation and question answering. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7856–7870, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[50] Zhiting Hu and Li Erran Li. A causal lens for controllable text generation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, Advances in Neural Information Processing Systems, volume 34, pages 24941–24955. Curran Associates, Inc., 2021.

[51] Zhiting Hu, Zichao Yang, Russ R Salakhutdinov, LIANHUI Qin, Xiaodan Liang, Haoye Dong, and Eric P Xing. Deep generative models with learnable knowledge constraints. Advances in Neural Information Processing Systems, 31, 2018.

[52] Kung-Hsiang Huang, Sam Tang, and Nanyun Peng. Document-level entity-based extraction as template generation. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5257–5269, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[53] Luyang Huang, Lingfei Wu, and Lu Wang. Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5094–5107, Online, July 2020. Association for Computational Linguistics.

[54] Pere-Lluís Huguet Cabot and Roberto Navigli. REBEL: Relation extraction by end-to-end language generation. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 2370–2381, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[55] Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. In AAAI, 2021.

[56] Harsh Jhamtani and Peter Clark. Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 137–150, Online, November 2020. Association for Computational Linguistics.

[57] Haozhe Ji, Pei Ke, Shaohan Huang, Furu Wei, Xiaoyan Zhu, and Minlie Huang. Language generation with multi-hop reasoning on commonsense knowledge graph. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 725–736, Online, November 2020. Association for Computational Linguistics.

[58] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. UNIFIEDQA: Crossing format boundaries with a single QA system. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1896–1907, Online, November 2020. Association for Computational Linguistics.

[59] Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Text modular networks: Learning to decompose tasks in the language of existing models. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1264–1279, Online, June 2021. Association for Computational Linguistics.

[60] Hyunwoo Kim, Byeongchang Kim, and Gunhee Kim. Will I sound like me? improving persona consistency in dialogues through pragmatic self-consciousness. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 904–916, Online, November 2020. Association for Computational Linguistics.

[61] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

[62] Karen Kukich. Design of a knowledge-based report generator. In 21st Annual Meeting of the Association for Computational Linguistics, pages 145–150, Cambridge, Massachusetts, USA, June 1983. Association for Computational Linguistics.

[63] Faisal Ladhak, Esin Durmus, He He, Claire Cardie, and Kathleen McKeown. Faithful or extractive? on mitigating the faithfulness-abstractiveness trade-off in abstractive summarization. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1410–1421, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[64] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[65] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 9459–9474. Curran Associates, Inc., 2020.

[66] Chenguang Li, Jiaqing Liang, Yanghua Xiao, and Haiyun Jiang. Towards fine-grained concept generation. IEEE Transactions on Knowledge and Data Engineering, pages 1–1, 2021.

[67] Haoran Li, Junnan Zhu, Jiajun Zhang, and Chengqing Zong. Ensure the correctness of the summary: Incorporate entailment knowledge into abstractive sentence summarization. In Proceedings of the 27th International Conference on Computational Linguistics, pages 1430–1441, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[68] Qing Li, Qingyi Tao, Shafiq Joty, Jianfei Cai, and Jiebo Luo. Vqa-e: Explaining, elaborating, and enhancing your answers for visual questions. ECCV, 2018.

[69] Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. CommonGen: A constrained text generation challenge for generative commonsense reasoning.

In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1823–1840, Online, November 2020. Association for Computational Linguistics.

[70] Emmy Liu, Chenxuan Cui, Kenneth Zheng, and Graham Neubig. Testing the ability of language models to interpret figurative language. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4437–4452, Seattle, United States, July 2022. Association for Computational Linguistics.

[71] Haochen Liu, Jamell Dacon, Wenqi Fan, Hui Liu, Zitao Liu, and Jiliang Tang. Does gender matter? towards fairness in dialogue systems. In Proceedings of the 28th International Conference on Computational Linguistics, pages 4403–4416, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[72] Pengfei Liu, Jinlan Fu, Yang Xiao, Weizhe Yuan, Shuaichen Chang, Junqi Dai, Yixin Liu, Zihuiwen Ye, and Graham Neubig. ExplainaBoard: An explainable leaderboard for NLP. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, pages 280–289, Online, August 2021. Association for Computational Linguistics.

[73] Ruibo Liu, Guoqing Zheng, Shashank Gupta, Radhika Gaonkar, Chongyang Gao, Soroush Vosoughi, Milad Shokouhi, and Ahmed Hassan Awadallah. Knowledge infused decoding. In International Conference on Learning Representations, 2022.

[74] Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S. Yu. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. Proceedings of the AAAI Conference on Artificial Intelligence, 35(7):6418–6425, May 2021.

[75] Zihan Liu, Mostofa Patwary, Ryan Prenger, Shrimai Prabhumoye, Wei Ping, Mohammad Shoeybi, and Bryan Catanzaro. Multi-stage prompting for knowledgeable dialogue generation. In Findings of the Association for Computational Linguistics: ACL 2022, pages 1317–1337, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[76] Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4288–4299, Online, June 2021. Association for Computational Linguistics.

[77] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8086–8098, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[78] Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 2795–2806, Online, August 2021. Association for Computational Linguistics.

[79] Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. Unified structure generation for universal information extraction. In Proceedings of the 60th Annual Meeting

of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5755–5772, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[80] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1906–1919, Online, July 2020. Association for Computational Linguistics.

[81] Kathleen McKeown. Text generation. Cambridge University Press, 1992.

[82] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. Cgmh: Constrained sentence generation by metropolis-hastings sampling. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01):6834–6842, Jul. 2019.

[83] George A. Miller. Wordnet: A lexical database for english. Commun. ACM, 38(11):39–41, nov 1995.

[84] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, Advances in Neural Information Processing Systems, 2022.

[85] Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, RISHITA ANUBHAI, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. Structured prediction as translation between augmented natural languages. In International Conference on Learning Representations, 2021.

[86] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander 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, Hong Kong, China, November 2019. Association for Computational Linguistics.

[87] Xinyu Pi, Wanjun Zhong, Yan Gao, Nan Duan, and Jian-Guang Lou. Logigan: Learning logical reasoning via adversarial pre-training. arXiv preprint arXiv:2205.08794, 2022.

[88] Matt Post and David Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. 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 1314–1324, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[89] Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. Glancing transformer for non-autoregressive neural machine translation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1993–2003, Online, August 2021. Association for Computational Linguistics.

[90] Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark, and Yejin Choi. Counterfactual story reasoning and generation. 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 5043–5053, Hong Kong, China, November 2019. Association for Computational Linguistics.

[91] Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. Cold decoding: Energy-based constrained text generation with langevin dynamics. arXiv preprint arXiv:2202.11705, 2022.

[92] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI Blog, 1(8):9, 2019.

[93] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67, 2020.

[94] Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4932–4942, Florence, Italy, July 2019. Association for Computational Linguistics.

[95] Vikas Raunak, Arul Menezes, and Marcin Junczys-Dowmunt. The curious case of hallucinations in neural machine translation. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1172–1183, Online, June 2021. Association for Computational Linguistics.

[96] Ehud Reiter. Natural language generation challenges for explainable ai. arXiv preprint arXiv:1911.08794, 2019.

[97] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pages 1135–1144, 2016.

[98] Gaetano Rossiello, Nandana Mihindukulasooriya, Ibrahim Abdelaziz, Mihaela Bornea, Alfio Gliozzo, Tahira Naseem, and Pavan Kapanipathi. Generative relation linking for question answering over knowledge bases. In International Semantic Web Conference, pages 321–337. Springer, 2021.

[99] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. 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 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics.

[100] Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In International Conference on Learning Representations, 2020.

[101] Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. Mixture models for diverse machine translation: Tricks of the trade. In International conference on machine learning, pages 5719–5728. PMLR, 2019.

[102] Xinyao Shen, Jiangjie Chen, Jiaze Chen, Chun Zeng, and Yanghua Xiao. Diversified query generation guided by knowledge graph. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, WSDM '22, page 897–907, New York, NY, USA, 2022. Association for Computing Machinery.

[103] Xinyao Shen, Jiangjie Chen, and Yanghua Xiao. Diversified paraphrase generation with commonsense knowledge graph. In Lu Wang, Yansong Feng, Yu Hong, and Ruifang He, editors, Natural Language Processing and Chinese Computing, pages 353–364, Cham, 2021. Springer International Publishing.

[104] Weiyan Shi, Yu Li, Saurav Sahay, and Zhou Yu. Refine and imitate: Reducing repetition and inconsistency in persuasion dialogues via reinforcement learning and human demonstration. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 3478–3492, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[105] Zhan Shi, Hui Liu, and Xiaodan Zhu. Descriptive image captioning with salient retrieval priors. In Luiza Antonie and Pooya Moradian Zadeh, editors, Proceedings of the 34th Canadian Conference on Artificial Intelligence, Canadian AI 2021, online, May 2021. Canadian Artificial Intelligence Association, 2021.

[106] Zhan Shi, Hui Liu, and Xiaodan Zhu. Enhancing descriptive image captioning with natural language inference. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 269–277, Online, August 2021. Association for Computational Linguistics.

[107] Chang Shu, Yusen Zhang, Xiangyu Dong, Peng Shi, Tao Yu, and Rui Zhang. Logic-consistency text generation from semantic parses. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4414–4426, Online, August 2021. Association for Computational Linguistics.

[108] Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Unsupervised commonsense question answering with self-talk. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4615–4629, Online, November 2020. Association for Computational Linguistics.

[109] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. Proceedings of the AAAI Conference on Artificial Intelligence, 31(1), Feb. 2017.

[110] Mingming Sun, Xu Li, and Ping Li. Logician and orator: Learning from the duality between language and knowledge in open domain. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2119–2130, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[111] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, page 3319–3328. JMLR.org, 2017.

[112] Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. Lexically constrained neural machine translation with Levenshtein transformer. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3536–3543, Online, July 2020. Association for Computational Linguistics.

[113] Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3621–3634, Online, August 2021. Association for Computational Linguistics.

[114] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations. In International Conference on Learning Representations, 2019.

[115] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. arXiv preprint arXiv:2201.08239, 2022.

[116] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. 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 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[117] Lindsey Vanderlyn, Gianna Weber, Michael Neumann, Dirk Väth, Sarina Meyer, and Ngoc Thang Vu. "it seemed like an annoying woman": On the perception and ethical considerations of affective language in text-based conversational agents. In Proceedings of the 25th Conference on Computational Natural Language Learning, pages 44–57, Online, November 2021. Association for Computational Linguistics.

[118] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

[119] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.

[120] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. Communications of the ACM, 57(10):78–85, 2014.

[121] Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. Heterogeneous graph neural networks for extractive document summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6209–6219, Online, July 2020. Association for Computational Linguistics.

[122] Qingyun Wang, Lifu Huang, Zhiying Jiang, Kevin Knight, Heng Ji, Mohit Bansal, and Yi Luan. PaperRobot: Incremental draft generation of scientific ideas. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1980–1991, Florence, Italy, July 2019. Association for Computational Linguistics.

[123] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171, 2022.

[124] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. Transactions on Machine Learning Research, 2022. Survey Certification.

[125] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, Advances in Neural Information Processing Systems, 2022.

[126] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1):36–45, 1966.

[127] Sarah Wiegreffe and Ana Marasovic. Teach me to explain: A review of datasets for explainable natural language processing. In J. Vanschoren and S. Yeung, editors, Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks, volume 1, 2021.

[128] Ludwig Wittgenstein. The blue and brown books, volume 34. Blackwell Oxford, 1958.

[129] Sixing Wu, Ying Li, Dawei Zhang, Yang Zhou, and Zhonghai Wu. Diverse and informative dialogue generation with context-specific commonsense knowledge awareness. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5811–5820, Online, July 2020. Association for Computational Linguistics.

[130] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144, 2016.

[131] Mengzhou Xia, Anjalie Field, and Yulia Tsvetkov. Demoting racial bias in hate speech detection. In Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media, pages 7–14, Online, July 2020. Association for Computational Linguistics.

[132] Yijun Xiao and William Yang Wang. On hallucination and predictive uncertainty in conditional language generation. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 2734–2744, Online, April 2021. Association for Computational Linguistics.

[133] Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie-yan Liu. A survey on non-autoregressive generation for neural machine translation and beyond. arXiv preprint arXiv:2204.09269, 2022.

[134] Runxin Xu, Jun Cao, Mingxuan Wang, Jiaze Chen, Hao Zhou, Ying Zeng, Yuping Wang, Li Chen, Xiang Yin, Xijin Zhang, Songcheng Jiang, Yuxuan Wang, and Lei Li. Xiaomingbot: A Multilingual Robot News Reporter. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 1–8, Online, July 2020. Association for Computational Linguistics.

[135] Weijia Xu and Marine Carpuat. EDITOR: An Edit-Based Transformer with Repositioning for Neural Machine Translation with Soft Lexical Constraints. Transactions of the Association for Computational Linguistics, 9:311–328, 03 2021.

[136] Jian Yang, Gang Xiao, Yulong Shen, Wei Jiang, Xinyu Hu, Ying Zhang, and Jinghui Peng. A survey of knowledge enhanced pre-trained models. arXiv preprint arXiv:2110.00269, 2021.

[137] Xi Ye and Greg Durrett. The unreliability of explanations in few-shot in-context learning. arXiv preprint arXiv:2205.03401, 2022.

[138] Xuwang Yin and Vicente Ordonez. Obj2Text: Generating visually descriptive language from object layouts. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 177–187, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[139] Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. A survey of knowledge-enhanced text generation. ACM Computing Surveys (CSUR), 2022.

[140] Chun Zeng, Jiangjie Chen, Tianyi Zhuang, Rui Xu, Hao Yang, Qin Ying, Shimin Tao, and Yanghua Xiao. Neighbors are not strangers: Improving non-autoregressive translation under low-frequency lexical constraints. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5777–5790, Seattle, United States, July 2022. Association for Computational Linguistics.

[141] Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. A survey of controllable text generation using transformer-based pre-trained language models. arXiv preprint arXiv:2201.05337, 2022.

[142] Houyu Zhang, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. Grounded conversation generation as guided traverses in commonsense knowledge graphs. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2031–2043, Online, July 2020. Association for Computational Linguistics.

[143] Maosen Zhang, Nan Jiang, Lei Li, and Yexiang Xue. Language generation via combinatorial constraint satisfaction: A tree search enhanced Monte-Carlo approach. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1286–1298, Online, November 2020. Association for Computational Linguistics.

[144] Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. 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 3363–3369, Hong Kong, China, November 2019. Association for Computational Linguistics.

[145] Chunting Zhou, Jiatao Gu, and Graham Neubig. Understanding knowledge distillation in non-autoregressive machine translation. In International Conference on Learning Representations, 2020.

[146] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. arXiv preprint arXiv:2205.10625, 2022.

[147] Shaowen Zhou, Bowen Yu, Aixin Sun, Cheng Long, Jingyang Li, and Jian Sun. A survey on neural open information extraction: Current status and future directions. arXiv preprint arXiv:2205.11725, 2022.

[148] Caleb Ziems, Jane Yu, Yi-Chia Wang, Alon Halevy, and Diyi Yang. The moral integrity corpus: A benchmark for ethical dialogue systems. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3755–3773, Dublin, Ireland, May 2022. Association for Computational Linguistics.

College-Related Question Answering based on Knowledge Graph

Cheng Peng, Hao Jiang, Junnan Dong, Xiao Huang
Department of Computing, The Hong Kong Polytechnic University

Abstract

An effective conversational agent can facilitate student advising. Many question-answering (QA) systems have been developed, but college-related QA has yet to be explored. It is a challenging task since questions from college students are domain-specific, while related training data is not available. The topics of the college-related questions are diverse, including the university, the department, undergraduate/graduate programs, and course content. To bridge this gap, we propose an effective question-answering framework for The Hong Kong Polytechnic University and Computer Science. It integrates a rule-based module, a knowledge-graph-based module, and a content-based module. The rule-based module can answer simple questions actually based on pre-defined rules. Then, we manually construct a college-related knowledge graph (KG) from related websites, Wikipedia, and textbooks. We embed the KG and employ the entity/relation representations to predict the answers. Moreover, we utilize a pre-trained language model to answer difficult questions based on college-related corpora. We evaluate the proposed QA framework by employing fourteen postgraduate students. The results show that it can answer most real-world questions.

1 INTRODUCTION

Knowledge Graphs (KGs) are an effective graph data structure modeling relational information. They consist of real-world entities and each entity's relations, corresponding to the nodes and edges in KG respectively [1, 2]. In KGs, each edge with its head entity and tail entity constitutes a triple (fact), i.e., (head entity, predicate, tail entity) or (h, p, t). To facilitate downstream tasks like Question Answering (QA) with abundant knowledge in KGs, Question Answering over Knowledge Graph (KGQA) is proposed [3, 4]. The main problem setting of KGQA is to translate the inputted natural language question into structured queries, and return the entities or predicates from KG as the answer. For example, given the question "How tall is YAO Ming?", the KGQA will translate this question to the query "(YAO Ming, Height, ?)", and return the answer by identifying this query's corresponding fact in KG "(YAO Ming, Height, 2.29m)".

This paper adopts a knowledge graph on special domains as the knowledge source for answers. Unlike other traditional KGQAs, which employ public large-scale KG databases including Freebase [5], DBpedia [9], and YAGO [10], this paper mainly targets to provide QA service to college students, answering the questions related to Hong Kong Polytechnic University (PolyU) and Computer-Science (CS), while triple facts for these two college-related domains are not included in the existing public knowledge graphs. To construct the corresponding domain

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

knowledge graph, we are required to collect domain information for the Hong Kong Polytechnic University and Computer-Science domains from Wikipedia, textbooks, and related websites.

However, some collected entities/predicates may have names different from those stored in KG, so translating natural language questions into KG queries needs to identify many different expressions in natural language questions [6, 7]. To solve this problem, we apply knowledge graph embedding [7, 8], which can learn a vector representation for each entity and predicate in the knowledge graph. Similar entities or predicates will have similar vectors. Although this way is good at matching similar entities, when there are misspelled words on entities extracted from questions, the performance is affected significantly. Hence, to make the Knowledge Embedding based Knowledge Graph Question Answering (KEQA) more powerful, we also design an entity matching algorithm to help KEQA to correct those wrong words.

What's more, as the domains of inputted questions are always unbounded while any KG is far from completion [11], for example, an unseen question's entity/predicate not involved in KG, it means that KGQA could not answer all the questions. To alleviate this problem, we add a content-based Question Answering (content-based QA) component to the whole QA framework, which consists of an advanced model standing for bidirectional encoder representations from transformers, just like the BERT [12]. By finetuning, the pre-trained model can search for answers from the collected text materials. Although we have extracted some triples from those materials, there is still much remaining knowledge. Adding them to the framework can expand the answerable range, but the processing speed of content-based QA is relatively slow, so it is only used as an alternative scheme. In addition, considering answering some simple questions immediately rather than using time-consuming machine learning models, the whole QA framework also includes a rule-based Question Answering (rule-based QA) component to improve the user experience and reduce the workload of the KGQA component.

In summary, rather than just relying on the advanced KGQA, the whole QA framework includes three different kinds of QA components. We design the rules for switching each QA component in our framework, which can make good use of every QA component. Eventually, we have a powerful QA framework in college-related domains. Our main contributions are presented as follows.

- We deliver a QA framework for college-related questions, including three different kinds of QA components, which make the whole QA framework powerful.
- We define the rules to switch each QA component in the whole QA framework.
- A domain knowledge graph is constructed which covers the PolyU and Computer-Science domains.
- We achieve performance improvement of the traditional KGQA by adding knowledge graph embedding and entity matching algorithm.

2 METHODOLOGY

The whole QA framework in this paper has three components: rule-based Question Answering (rule-based QA), Question Answering over Knowledge Graph (KGQA), and content-based Question Answering (content-based QA).

2.1 Overall Architecture of the Proposed Framework

The whole QA framework is mainly relying on the KGQA. After constructing more than 15000 Hong Kong Polytechnic University (PolyU) and Computer-Science (CS) related triples, this huge PolyU and CS knowledge graph could cover most of the user's questions in these two domains. Especially with the improvement of knowledge graph embedding [7, 8] and entity matching algorithm, the KGQA component can deal with those questions with different expressions in entities or predicates.

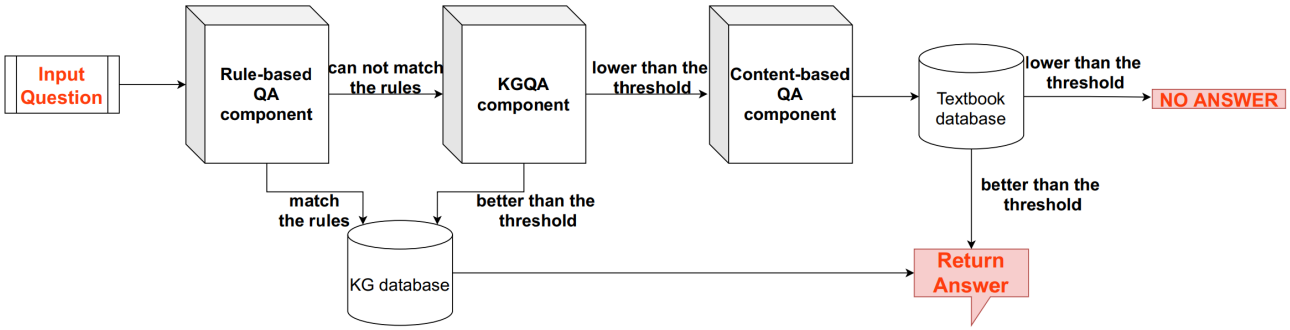


Figure 1: The architecture of the proposed question answering framework.

However, KGQA still faces two challenges: (i) time-consuming for KG reasoning and (ii) knowledge-agnostic to those questions that KG does not cover.

As KGQA employs language models (three language models in this paper's KGQA component) to process questions, it is relatively time-consuming. For those questions with simple grammatical structure and the same expression of entity in the knowledge graph, there is no necessity to use KGQA models. For example, given the question "Where is library?", assuming that "library" is an entity as same as the "library" entity expression in the knowledge graph, it is empirically a question about locations, so the query in KG will be "(Library, Location, ?)". To answer these one-hop questions without using language models, the rule-based QA component is proposed. By matching manually-created regular expressions, the rule-based QA component could answer a lot of questions without using language models, which is much quicker than the KGQA component.

For those triples that the knowledge graph doesn't cover, KGQA could not answer the questions related to them. For instance, KGQA could not answer the question "What's the use of loss function?" if there is not a triple like "(Loss Function, Use, xxx)" in the knowledge graph. Though we collect many documents about PolyU and CS, there is still much knowledge that we do not extract from the textbooks. Introducing a content-based QA component could facilitate QA by making full use of text material in the database. After adding the content-based QA component, the QA framework could answer more questions than that only based on KG.

The overall architecture of our proposed QA system is illustrated in Figure 1. The rule-based QA component is added before the KGQA component. It will try to answer the question first without using time-consuming language models. The content-based QA component is added after the KGQA component, for answering those questions that KGQA is not able to answer but answerable in the textbook database.

The basic switching rules for each QA component are as follows. Initially, the inputted question will be delivered to the rule-based QA component to check whether there is any regular expression rule matched. If the question could not find the corresponding regular expression rule, it will be sent to the KGQA component. If KGQA could not answer the question (two cases: KGQA could not find the corresponding head entity in the knowledge graph; the distance between the predicted triple embedding representation and its nearest triple in candidates is larger than the pre-defined distance threshold), the question will be sent to content-based QA component. If the question could not be answered by content-based QA (the output score is lower than the pre-defined score threshold), it means that the whole QA framework is unable to answer this question.

2.2 Construction of College-related Knowledge Graph

The methods of constructing PolyU and CS related knowledge graphs in this paper are shown as follows. (i) Manually browsing or using python crawler technology to collect knowledge on the official websites, such as Hong Kong Polytechnic University (PolyU) main page. After equipping with the knowledge, we transform the knowledge into triples like "(VA Student Canteen, Location, G/F, Shaw Amenities Building (Core VA))". (ii)

Using Wikipedia (a Python library)[1] to access and parse data from Wikipedia. Similar to the first method, we transform the knowledge to a triple format [1, 2]. (iii) Manually collecting some textbooks and using a pre-trained reading comprehension model [13] to search the corresponding tail entity that we want. For example, we use the reading comprehension model to search for the answer of "Who is the inventor of JAVA?" from JAVA textbook. If the returned text is "James Gosling", which is a correct answer, we will transform it to a triple "(JAVA, inventor, James Gosling)".

Based on the above three methods, there are more than 15,000 PolyU and CS related triples constructed in total, which can answer a huge range of Hong Kong Polytechnic University and Computer-Science questions.

2.3 Rule-based Question Answering

The first question answering component in this paper is the rule-based Question Answering (rule-based QA) component. This kind of approach is one of the initial methods used by traditional QA systems [24], which uses different kinds of rules, e.g., semantic rules and keywords, to understand the intentions of inputted questions. Furthermore, it is also used to assist other types of QAs, especially KGQA. For example, Z. Jiang et al. built a medical lexicon to classify and query questions in KGQA through rule-based matching methods and string-matching algorithms [25].

Similarly, the rule-based QA component in our paper extracts entities and predicates from questions based on the string matching algorithms. By matching the questions with pre-defined regular expressions, we could get the head entities and predicates in questions without using any time-consuming models.

However, due to the limitations of labor-consuming rule mining, we merely use the rule-based component to process two types of questions. The first one is simple and common questions, such as "Where is library?", which is used for increasing the whole QA framework processing speed. When the component receives a question, it will initially check whether the question contains pre-defined question words in the rule dictionary for the first classification which aims to recognize the general intention. In the example above, the question will trigger the "Where" rule for locations. Then, according to the recognized intention, the question will be allocated to the special rule dictionary for the second classification in which we design many keywords for regex matching. If the question matches any regular expression, the corresponding predicate will be determined. Consequently, all the entities related to the predicate in our database will be identified. Finally, based on Levenshtein Distance, we calculate the distance between the head entity candidates and the text except for pre-defined rule keywords in question. The closest head entity candidate will be regarded as the final head entity result. The tail entity obtained by the head entity and predicate in the knowledge database is the final answer to the question. The second type is questions that need some simple operations after detecting predicates and head entities rather than just returning the tail entity as the answer, such as the summation in "How many elective courses in MSc. IT?". Under such circumstances, the predicate and head entity detection is the same as that for the first type. The only difference is that processing the second question type requires additional operation after head entity and predicate recognition. For different operations, we formulate different rules in the rule-based QA component, such as counting the number of tail entities. For the example question, after recognizing "Elective Courses" as the predicate and "IT MSc" as the head entity, calculating the number of tail entities in the triple pattern "(IT MSc, Elective Courses, ?)" is necessary for obtaining the final answer. Therefore, we set the rule that after recognizing the intention of "Number" and the predicate of "Elective Courses", the rule-based QA component needs to calculate the number of elective courses.

2.4 Question Answering based on Knowledge Graph Embedding

In our framework, in order to better identify the head entity and predicate by recognizing different expressions of entity/predicate in natural language questions [6, 7] correctly, the knowledge graph embedding [7, 8] is applied to

[1] `https://pypi.org/project/wikipedia/`

the KGQA component, which can learn a vector representation for each predicate and entity in knowledge graph and keep the original relations simultaneously. What's more, because there is a large number of triples in the knowledge graph, it's very expensive and noisy to compare the predicted triple with all the triples in KG database. Therefore, we introduce a head entity detection model which could effectively prioritize the candidate triples. For example, given the question "Where is VA canteen?", the detected head entity is "VA canteen". The candidate triples to be compared will only be the ones with the head entity "VA canteen". The main idea of the KGQA component based on knowledge graph embedding in this paper is illustrated in Figure 2.

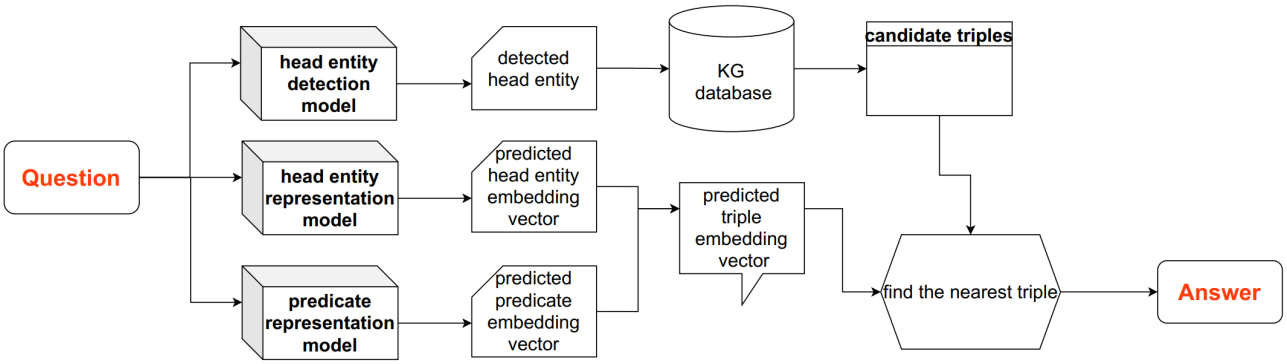


Figure 2: The workflow of the KGQA component.

Table 11: The notations in KGQA component and corresponding definitions.

Notations	Definitions
G	this framework's knowledge graph
Q	the questions in training set
(h, p, t)	a triple (fact) in G, (head entity, predicate, tail entity)
d	dimension of entity/predicate embedding representation
$\mathbf{e}_h \in \mathbf{R}^{1*d}$	head entity embedding representation vector
$\mathbf{e}_p \in \mathbf{R}^{1*d}$	predicate embedding representation vector
$\mathbf{e}_t \in \mathbf{R}^{1*d}$	tail entity embedding representation vector
$\hat{\mathbf{e}}_h \in \mathbf{R}^{1*d}$	predicted head entity embedding representation vector
$\hat{\mathbf{e}}_p \in \mathbf{R}^{1*d}$	predicted predicate embedding representation vector
$\hat{\mathbf{e}}_t \in \mathbf{R}^{1*d}$	predicted tail entity embedding representation vector
HED	Head Entity Detection model
$\mathbf{HED}_{entity}$	detected head entity token in HED
$\mathbf{HED}_{non}$	detected non head entity token in HED

To better describe the workflow of KGQA, we summarize the significant notations in Table 11. The question answering process of the KGQA component is mainly five steps: (i) Input the question into the Head Entity Detection (HED) model to get $\mathbf{HED}_{entity}$ and $\mathbf{HED}_{non}$. (ii) Generate the candidate-triple set from the KG database by the detected head entity name $\mathbf{HED}_{entity}$. (iii) Input the question into the predicate representation model to get $\hat{\mathbf{e}}_p$. (iv) Input the question into the head entity representation model to get $\hat{\mathbf{e}}_h$. (v) For all triples in the candidate triple set, calculate each one's distance to the predicted triple ($\hat{\mathbf{e}}_h$, $\hat{\mathbf{e}}_p$, $\hat{\mathbf{e}}_t$), and return the nearest candidate triple as the answer.

There are two cases in which the KGQA component is not able to answer the questions. Firstly, in the above (ii) step, KGQA may not find the corresponding head entity in the knowledge graph. Secondly, in the above (v) step, the distance between the predicted triple ($\hat{\mathbf{e}}_h$, $\hat{\mathbf{e}}_p$, $\hat{\mathbf{e}}_t$) and its nearest triple in candidates may be larger than the pre-defined distance threshold. If one of the above cases happens, the question will be sent to the content-based QA component.

2.4.1 Head Entity Detection Model

In the first step of the KGQA component, there is a Head Entity Detection (HED) model which aims to find target tokens (words) in the question for the head entity, so that the candidate triples will be the triples with the same or similar detected head entity names, rather than all triples in KG. The predicted representation of head entity $\hat{\mathbf{e}}_h$ from the representation model is used for dealing with ambiguity.

We adopt the long short-term memory (LSTM) [14] structure, which is a bidirectional recurrent neural network to facilitate a concise KGQA module. Figure 3(a) shows the architecture of HED model.

Given a question with length L, the HED model will use a pre-trained model (GloVe [15]) to map it to an L-length sequence of word embedding vectors $\mathbf{x}_j$ (for j = 1, ..., L). Then, there is the bidirectional LSTM [14] structure to generate $\mathbf{h}_j$. After the LSTM structure, a fully connected layer and a softmax function are applied to $\mathbf{h}_j$, generating the final vector $\mathbf{v}_j \in \mathbf{R}^{(2*1)}$ for each token in this L-length sentence. The two values in $\mathbf{v}_j$ represent the probabilities that the $\mathbf{j}^{th}$ token belongs to the entity name token and non-entity name token respectively. If the probability of the entity name token is higher, it will be classified as $\mathbf{HED}_{entity}$. To sum up, HED model can classify each token in question whether it is a token in the head entity, and finally connect one or several tokens as the head entity name. We denote these tokens as $\mathbf{HED}_{entity}$, and the remaining tokens in question as $\mathbf{HED}_{non}$.

To train the Head Entity Detection (HED) model, this paper uses the questions in training set Q as training data input and tags their head entity names as the training data output. For instance, the training question "Where is VA canteen" is tagged as "O O I I" in which "O" stands for $\mathbf{HED}_{non}$, and "I" means $\mathbf{HED}_{entity}$. After training, the HED model can identify which tokens in question are $\mathbf{HED}_{entity}$.

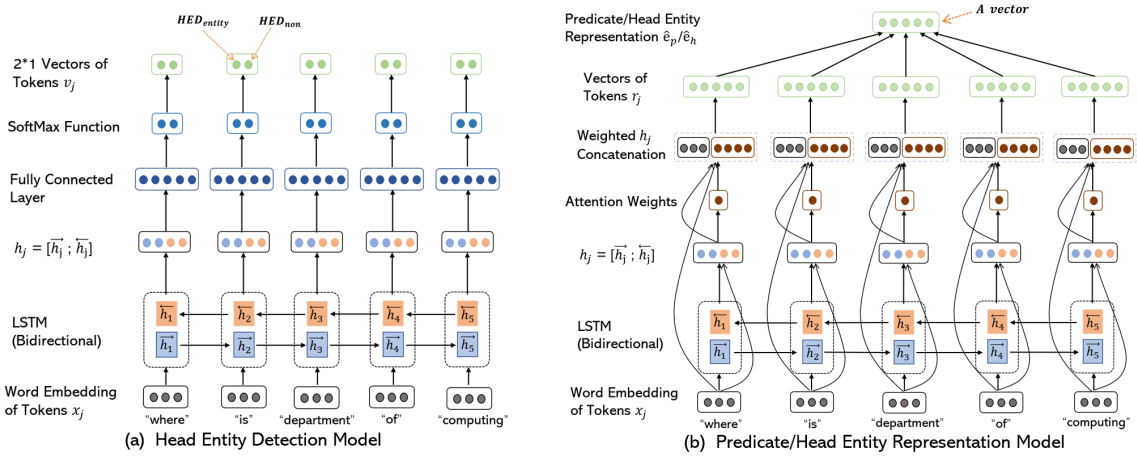


Figure 3: Architectures of head entity detection model and predicate/head entity representation model.

2.4.2 Knowledge Graph Embedding

Knowledge graph embedding [7, 8] could represent each predicate and entity in the knowledge graph as a vector, while the triple's original structure and relation in the knowledge graph are well-preserved in those embedding vectors. The reason why this paper uses knowledge embedding is that it can disambiguate those inputted questions' head entity name or predicate name which is different from the corresponding entity name or predicate name in the knowledge graph database. The traditional KGQA is hard to deal with this issue while Knowledge Embedding based Knowledge Graph Question Answering (KEQA) could potentially improve the question answering accuracy in this case.

There are many different kinds of knowledge graph embedding algorithms [7, 8], and the basic ideas of most knowledge graph embedding algorithms [11, 16, 17] are similar. For each triple (h, p, t) in knowledge graph

G, denoting its knowledge graph embedding representation as ($\mathbf{e}_h$, $\mathbf{e}_p$, $\mathbf{e}_t$), which are three vectors for h, p, t respectively. The knowledge graph embedding algorithm will initialize $\mathbf{e}_h$, $\mathbf{e}_p$ and $\mathbf{e}_t$ randomly [11, 16] (or use word embedding model to initialize [18, 19]). In addition, each knowledge graph embedding algorithm has a unique function f (·), which could represent the tail entity vector by head entity vector and predicate vector in the same triple ($\mathbf{e}_t \approx$ f ($\mathbf{e}_h$, $\mathbf{e}_p$)). Finally, the embedding algorithm's target is to minimize the distance between $\mathbf{e}_t$ and f ($\mathbf{e}_h$, $\mathbf{e}_p$) for all the triples in the knowledge graph G by training, while the training step is using both positive and negative samples in KG. Each knowledge graph embedding algorithm is mainly different in the function f (·).

In this paper, TransE [16], a typical knowledge graph embedding method is picked, which defines the "$\mathbf{e}_t \approx \mathbf{e}_h + \mathbf{e}_p$" relation (function f (·)) for each triple ($\mathbf{e}_h$, $\mathbf{e}_p$, $\mathbf{e}_t$) in KG. We apply TransE to initialize the values of $\mathbf{e}_h$, $\mathbf{e}_p$ and $\mathbf{e}_t$ randomly at the first stage, and then minimize the overall distance between $\mathbf{e}_t$ and f ($\mathbf{e}_h$, $\mathbf{e}_p$) by training. After training, we obtain the embedding vectors of the head entity and predicate in one specific triple, which are $\mathbf{e}_h$ and $\mathbf{e}_p$; then, we can calculate the representation vector of the tail entity ($\mathbf{e}_t$) in this triple automatically, which is $\mathbf{e}_h + \mathbf{e}_p$.

2.4.3 Predicate and Head Entity Representation Models

The target of predicate and head entity representation models is outputting the predicate representation vector $\hat{\mathbf{e}}_p$ and head entity representation vector $\hat{\mathbf{e}}_h$ in the knowledge graph embedding space by the given question. The input of the representation model is a natural language question, and the output is a vector.

For the representation model, to keep neural network architecture simple, similar to the previous HED model, the representation model mainly consists of a bidirectional long short-term memory (LSTM) structure [14]. The part difference with the HED model is that the representation model's output (head entity/predicate representation vector) is related to every token in question, so an attention layer is added, which could take the order and the importance of all words into consideration.

The predicate and head entity representation models' architecture is shown in Figure 3(b). The workflow before the attention layer (given a question with length L) is similar to that of the HED model. After the attention layer and concatenation, each token will generate a vector $\mathbf{r}_j$ (for j = 1, . . . , L). Finally, the predicted predicate/head entity representation vector is computed by the mean of all $\mathbf{r}_j$, which is $\hat{\mathbf{e}}_p/\hat{\mathbf{e}}_h = \frac{1}{L}\sum_{j=1}^{L} \mathbf{r}_j^\top$.

To train the predicate and head entity representation models, we need to label each question's corresponding predicate/head entity representation vector $\mathbf{e}_p$/$\mathbf{e}_h$. After training, the predicted output vector $\hat{\mathbf{e}}_p$/$\hat{\mathbf{e}}_h$ of inputted question will be close to this question's real predicate/head entity knowledge graph embedding representation vector $\mathbf{e}_p$/$\mathbf{e}_h$.

2.4.4 Joint Search on Embedding Spaces

After obtaining the candidate triples from the HED model as well as predicted predicate representation $\hat{\mathbf{e}}_p$ and predicted head entity representation $\hat{\mathbf{e}}_h$ from corresponding representation models, the next step of the KGQA component is to find the triple from all the candidates which best matches the predicted $\hat{\mathbf{e}}_p$ and $\hat{\mathbf{e}}_h$.

Let C be the set of candidate triples generated from the HED model. In order to calculate the distance between each candidate triple (h, p, t) and the predicted triple representation vector ($\hat{\mathbf{e}}_h$, $\hat{\mathbf{e}}_p$, $\hat{\mathbf{e}}_t$), this framework makes full use of the relation $\mathbf{e}_t \approx$ f ($\mathbf{e}_h$, $\mathbf{e}_p$) = ($\mathbf{e}_h$ + $\mathbf{e}_p$) in TransE [16], which means that the predicted tail entity representation $\hat{\mathbf{e}}_t$ could be computed by predicted predicate representation $\hat{\mathbf{e}}_p$ and predicted head entity representation $\hat{\mathbf{e}}_h$. Thus, a direct way to compute the distance between the predicted triple vector and each triple embedding representation vector ($\mathbf{e}_h$, $\mathbf{e}_p$, $\mathbf{e}_t$) in candidate triples is achieved by the sum distance of $\hat{\mathbf{e}}_h$ and $\mathbf{e}_h$, $\hat{\mathbf{e}}_p$ and $\mathbf{e}_p$, $\hat{\mathbf{e}}_t$ and $\mathbf{e}_t$. Moreover, as the real final-selected head entity name in C is likely to be the same as $\mathbf{HED}_{entity}$, and the predicate name is also likely to be mentioned in the other text of the question ($\mathbf{HED}_{non}$)

[20], the joint distance metric is proposed as follows.

$$\underset{(h,p,t)\in C}{\text{minimize}} \left\|\mathbf{e}_p - \hat{\mathbf{e}}_p\right\|_2 + \beta_1 \left\|\mathbf{e}_h - \hat{\mathbf{e}}_h\right\|_2 + \beta_2 \left\|\mathbf{e}_t - (\hat{\mathbf{e}}_h + \hat{\mathbf{e}}_p)\right\|_2 - \beta_3 \operatorname{sim}\left[h, \mathbf{HED}_{\text{entity}}\right] - \beta_4 \operatorname{sim}\left[p, \mathbf{HED}_{\text{non}}\right]. \tag{15}$$

The first three terms are to calculate the distance between one candidate triple (h, p, t) and the predicted triple representation vector ($\hat{\mathbf{e}}_h$, $\hat{\mathbf{e}}_p$, $\hat{\mathbf{e}}_t$) while using $\mathbf{l}^2$ norm to measure the distance in embedding space. The final two terms are to check whether the head entity name (h) and predicate name (p) in candidate triple are mentioned in the question. Function sim() is used to calculate the string similarity. $\beta 1$, $\beta 2$, $\beta 3$, and $\beta 4$ are pre-defined weights [20].

2.4.5 Entity Detection Improvement

According to the workflow of the KGQA component, it's clear that generating the candidate triples by the detected head entity is important. After inputting the question into the head entity detection (HED) model, the HED model will judge which tokens in question constitute the head entity. However, there are three possible problems in this part, including the detected head entity containing some other tokens besides the correct head entity name, the detected head entity only being a part of the correct head entity name, and the detected head entity having some misspelled words. Any of these situations could cause entity matching to fail. If the detected head entity could not match any head entity in the knowledge graph, there will be no candidate triples. Correspondingly, the joint search in embedding spaces cannot play its role. Therefore, it is necessary to deal with the above three cases by entity detection improvement based on some string-matching algorithms.

For the first case, the detected head entity may include some unrelated tokens besides the original head entity name. For example, the detected head entity is "the department of computing in PolyU" while the head entity name stored in KG is "department of computing". In this case, we use N-Gram [21] algorithm to extract 1-gram, 2-gram to n-gram from each fragment in detected entity words, which will form a set of n-gram. Then, the framework will use the n-gram set to find head entities in the KG dataset, generating a set of detected head entity candidates. The remaining issues are detected head entity that is merely a part of the head entity name in KG or includes misspelled words. For instance, the correct head entity name in KG is "department of computing", while the detected head entity is "computing department" (a part of the correct head entity) or "depart of computing" (misspelled words). In the circumstances, the KGQA component is difficult to accurately generate the candidate triples from these two kinds of detected head entities. To match these two kinds of "wrong-input" head entity names in question, a string-matching algorithm named Monge and Elkan (Mon-Elk) based on Levenshtein [22] is added. It is a hybrid string-matching algorithm at character and token levels, which will calculate the similarity between each token in different sentences. The formula is shown in Eq. (16). A and B are the strings that we want to calculate how similar they are, and A_i, B_j are the tokens in A and B respectively.

$$\operatorname{match}(A, B) = \frac{1}{|A|} \sum_{i=1}^{|A|} \max_{j=1}^{|B|} \operatorname{match}\left(A_i, B_j\right). \tag{16}$$

Following prevailing studies [23], n-gram combined with set-matching methods is the best matching algorithm for the matching problem. However, the experiment is done on data with no typing errors. Compared with the best method, the second best one, Mon-Elk based on Levenshtein, is more suitable for our applications, because Levenshtein edit distance can better determine the extent of spelling errors. After calculating the Mon-Elk similarity between the detected head entity and each entity in KG, the knowledge embedding based KGQA component will select those entities with a similarity larger than the pre-defined similarity threshold as head entity candidates and use them to generate the candidate triples.

2.5 Content-based Question Answering

The final component of our framework is content-based question answering (content-based QA). All the documents in this part are crawled by us from textbooks or through Wikipedia API. If this component is not able to provide the answer, the whole QA framework will return "no answer" as the final reply to users. In this part, we divide the content-based question answering component into two aspects, document retrieval and content reader. The general workflow of content-based QA is illustrated in Figure 4. In this paper, we deploy the content-based question answering component through an open end-to-end framework, haystack[2].

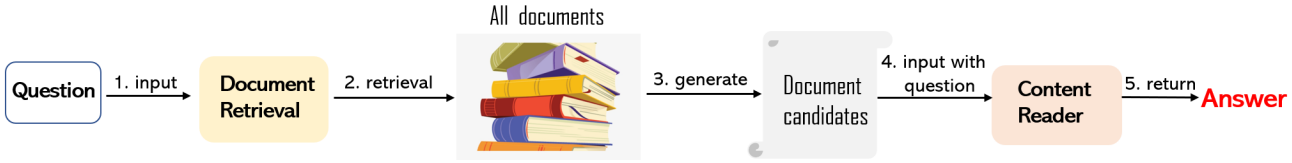


Figure 4: The workflow of content-based question answering component.

2.5.1 Document Retrieval

Generally, dense retrievers perform better than sparse methods, like TF-IDF and BM25, but dense approaches are computationally more expensive, especially during indexing [27]. Therefore, to make our framework more simplified and low computational complexity, we only perform the retrieval through a simple method, Term Frequency-Inverse Document Frequency (TF-IDF) which is a commonly used baseline for information retrieval. There are two main ideas of TF-IDF. Firstly, the more words the two documents overlap, the more relevant they are. In addition, the fewer documents a word is contained in, the more important the word is.

When the component receives a question, all the documents will be calculated TF-IDF score with it and sorted from largest to smallest according to the score. The top n documents will be prioritized as document candidates in which the content reader searches for answers. In our framework, we value the top two as the document candidates.

2.5.2 Content Reader

After retrieving the documents, the content reader takes a question and a set of candidate documents as inputs and returns an answer by selecting a text span within the documents. For selecting the reader model in this part, we choose an easy-to-deploy and powerful model – RoBERTa [26].

As it is expensive to manually annotate the span of documents and valid texts to train the content reader, we apply the pre-trained RoBERTa model by Deepset[3] on SQuAD2.0 dataset which includes both answerable and unanswerable questions. Compared with the model trained on SQuAD1.0/1.1 dataset, there is an additional binary classifier to automatically identify whether a question can be answered or not. The span predictor and classifier are trained jointly [26]. To fine-tune the model for our tasks, we create some question-answer pairs in our documents.

In terms of the process of the content-based QA component, the model takes a question and a set of candidate documents obtained by the document retrieval as input, returning a set of answers with a predicted score and no-answer score of the query. In our framework, we set 0 as the threshold of the no-answer score. If the no-answer score is higher than or equal to 0, it means that the question can be answered. In this case, the content-based QA component will return the result with the highest predicted score as the final answer. On the contrary,

[2] https://github.com/deepset-ai/haystack
[3] https://huggingface.co/deepset/roberta-base-squad2/

if the no-answer score is lower than the threshold, the question will be considered unanswerable. Since the content-based question answering module is the final part of our framework, when the query is considered unanswerable, the QA framework will return a no-answer signal to users directly.

3 EXPERIMENTS

After implementing the whole QA framework which includes three different QA components mentioned above, we design a website and embed the QA service interface for front-end users. After the users log in, they could talk with the chatbot for answers.

To evaluate the effectiveness of this proposed QA framework, we invite 14 students in the department of computing at PolyU and collect a total of 730 questions related to PolyU or Computer-Science. Since we regulate those testers to ask questions about PolyU and Computer-Science but do not limit the candidate concepts (entity names) and relations they could ask, there are some unbounded questions that our QA framework is not able to answer (no corresponding knowledge graph or text material in the database). Among all the 730 inputted questions, there are 638 valid questions. The term "valid" means that the knowledge graph database or text corpus in this paper is enough to deal with this specific question. So, for all these 14 students, our QA framework is possible to answer 87.4% of all testers' questions. It means that this framework's database is capable to satisfy most general college students' needs.

Table 12: The accuracy of our question answering framework.

	QA Framework Accuracy
PolyU domain	87.1%
CS domain	80.6%
PolyU & CS domain	83.6%

Table 2 shows the accuracy of the whole QA framework in this paper. For all those 638 valid questions, the QA framework could correctly answer 534 questions, which is up to 83.6%. In addition, we also calculate the accuracies in PolyU and Computer-Science domains respectively: For questions related to Computer-Science domain, the accuracy is lower than the PolyU domain, which is mainly because the Computer-Science domain questions are more complex and multifaceted. They are more suitable to be processed by content-based QA component. For PolyU related questions, the KGQA component is sufficient to answer most of them, and we achieve relatively satisfying performance.

Although our proposed framework obtains good accuracy for answering college-related questions, there are still a lot of unbounded questions (12.6%). In other words, if we include all the unbounded questions, the overall accuracy is 73.1% (calculated by Eq. (17), N_{valid} means the number of all valid questions in testing questions and N_{all} means the number of all testing questions), which is far from perfect.

$$Overall\ Accuracy = \frac{N_{valid}}{N_{all}} * QA\ Framework\ Accuracy. \tag{17}$$

To keep improving the overall accuracy, the direct way is constructing more knowledge graphs or collecting more text materials to make this framework more powerful.

4 CONCLUSIONS AND FUTURE WORK

In this paper, our target is to design a QA framework that can answer the questions related to Hong Kong Polytechnic University (PolyU) and Computer-Science (CS). Inspired by the KGQA that allows users access knowledge in some public knowledge graph databases, we construct the PolyU and CS related knowledge graph with more than 15000 triples to facilitate knowledge reasoning. In order to handle the ambiguity of different expressions, misspelled words, and partial names in questions, this paper improves the KGQA component by using knowledge graph embedding and off-the-shelf string-matching algorithms like the Mon-Elk matching algorithm and N-Gram. The KGQA component in this paper could have a better performance than the traditional simple KGQA.

In addition to KGQA, a rule-based QA component and content-based QA component are proposed to enhance the whole framework. Rule-based QA could answer some simple questions based on logical rules, while content-based QA is able to answer some unbounded questions that the knowledge graph doesn't cover but can be answered from rich texts.

After defining the rules for switching each QA component in the whole framework, we could make full use of each QA component and propose a powerful QA framework for answering college-related questions.

In future work, except for constructing more knowledge graphs and collecting more text materials, we will try to keep improving this QA framework through the following aspects. (i) We will apply more cutting-edge multi-hop knowledge graph reasoning algorithms for complex question answering. (ii) The continual learning ability for a conversational chatbot will be one of our main focuses to improve the usability of the framework.

References

[1] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In AAAI Conference on Artificial Intelligence, 2181–2187, 2016.

[2] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge Graph Embedding: A Survey of Approaches and Applications. IEEE Transactions on Knowledge and Data Engineering, 29, 12:2724–2743, 2017.

[3] Zihang Dai, Lei Li, and Wei Xu. CFO: Conditional Focused Neural Question Answering with Large-Scale Knowledge Bases. In Annual Meeting of the Association for Computational Linguistics, 800–810, 2016.

[4] Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. An End-to-End Model for Question Answering over Knowledge Base with Cross-Attention Combining Global Knowledge. Annual Meeting of the Association for Computational Linguistics, 221–231, 2017.

[5] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In International Conference on Management of Data, pages 1247–1250, 2008.

[6] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In Conference on Empirical Methods in Natural Language Processing, 1533–1544, 2013.

[7] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In ACL-IJCNLP, 1321–1331, 2015.

[8] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge Graph and Text Jointly Embedding. In Conference on Empirical Methods in Natural Language Processing, 1591–1601, 2014.

[9] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. DBpedia–A Large-Scale, Multilingual Knowledge Base Extracted From Wikipedia. Semantic Web 6, 2, 167–195, 2015.

[10] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A Core of Semantic Knowledge. In International World Wide Web Conference, 697–706, 2007.

[11] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In AAAI Conference on Artificial Intelligence, 2181–2187, 2015.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In NAACL-HLT, 4171–4186, 2019.

[13] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In Conference on Empirical Methods in Natural Language Processing, 2016.

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In International Conference on Learning Representations, 2015.

[15] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In Conference on Empirical Methods in Natural Language Processing, 1532–1543, 2014.

[16] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In Annual Conference on Neural Information Processing Systems, 2787–2795, 2013.

[17] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. SSP: Semantic Space Projection for Knowledge Graph Embedding with Text Descriptions. In AAAI Conference on Artificial Intelligence, 3104–3110, 2017.

[18] Teng Long, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. Leveraging Lexical Resources for Learning Entity Embeddings in Multi-Relational Data. In Annual Meeting of the Association for Computational Linguistics, 112–117, 2016.

[19] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning with Neural Tensor Networks for Knowledge Base Completion. In Annual Conference on Neural Information Processing Systems, 926–934, 2013.

[20] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge Graph Embedding Based Question Answering. In ACM International Conference on Web Search and Data Mining, 105–113, 2019.

[21] Zhaohui Chao, and Lin Li. The Combination of Context Information to Enhance Simple Question Answering. In International Conference on Behavioral, Economic, and Socio-Cultural Computing, 2018.

[22] Alvaro E. Monge, and Charles P. Elkan. The field matching problem: Algorithms and applications. In International Conference on Knowledge Discovery and Data Mining, 267–270, 1996.

[23] Najlah Gali, Radu Mariescu-Istodor, Damien Hostettler, and Pasi Fränti. Framework for syntactic string similarity measures. Expert Systems with Applications, Volume 129, 169–185, 2019.

[24] K.S.D. Ishwari, A.K.R.R. Aneeze, S. Sudheesan, H.J.D.A. Karunaratne, A. Nugaliyadde, and Y. Mallawarrachchi. Advances in natural language question answering: A review. arXiv preprint arXiv:1904.05276, 2019.

[25] Zhixue Jiang, Chengying Chi, and Yunyun Zhan. Research on Medical Question Answering System Based on Knowledge Graph. IEEE Access, 9, 21094–21101, 2021.

[26] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 , 2019.

[27] Negar Arabzadeh, Xinyi Yan, and Charles L. A. Clarke. Predicting Efficiency/Effectiveness Trade-offs for Dense vs. Sparse Retrieval Strategy Selection. In ACM International Conference on Information & Knowledge Management, 2862–2866, 2021.

Implicit Sentiment Analysis of Chinese Texts based on Contextual Information and Knowledge Enhancement

Zhenghui Cao[†] Siqi Wang[‡] Haofen Wang[‡ *] Wenqiang Zhang[†]
[†] Fudan University, Shanghai, China {caozh20,wqzhang}@fudan.edu.cn
[‡] Tongji University, Shanghai, China {siqi_wang,nobot}@tongji.edu.cn

1 Abstract

The implicit sentiment analysis task makes the traditional explicit sentiment analysis methods no longer applicable due to the obscure and ambiguous expression of the sentiment text it focuses on and the lack of sentiment cue words containing obvious sentiment tendencies. In this paper, we propose a sentiment analysis method incorporating contextual information and external common sense knowledge for implicit sentiment analysis. We first propose a retrieval method to extract valid sentiment triples from a knowledge base and build an adaptive fusion of external common sense knowledge embedding layers to extend the semantic information of implicit sentiment expressions. Multipolar orthogonal attention mechanism is then used to learn embedding representations for implicit sentiment expressions, and an attention layer that incorporates context is introduced to mine and combine valid information in the context. Finally, the common sense knowledge information provided by the external knowledge base is fused with the contextual representation and the semantic information of the sentiment expression itself to achieve an effective extension of the implicit sentiment sentence representation and to perform sentiment label prediction. Experimental results on the SMP2019-ECISA dataset show that the method outperforms existing state-of-the-art methods and can fully exploit contextual information and external common sense knowledge to effectively improve the analysis of implicit sentiment expressions.

2 Introduction

As one of the most popular research topics in the field of natural language processing, sentiment analysis tasks have been widely researched and applied in various fields such as academic research and product review mining, social opinion analysis, content-based recommendation, and so on. People's experience and sentimental expression of objective things are often subjective and diverse, either by giving direct subjective statements of opinions, or by using facts or rhetorical hints to ideas, that is, by using sentences that do not contain obvious sentimental cues can also express a subjective sentiment. In connection with real life, people actually use a lot of implicit expressions to convey their feelings. There are examples as shown in the Fig 1 .

Both of the above examples do not contain obvious sentimental cues, but they invariably convey the speaker's sentimental disposition. Sentiment expressed implicitly rather than using words with sentimental colors are called

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[*]*Corresponding author

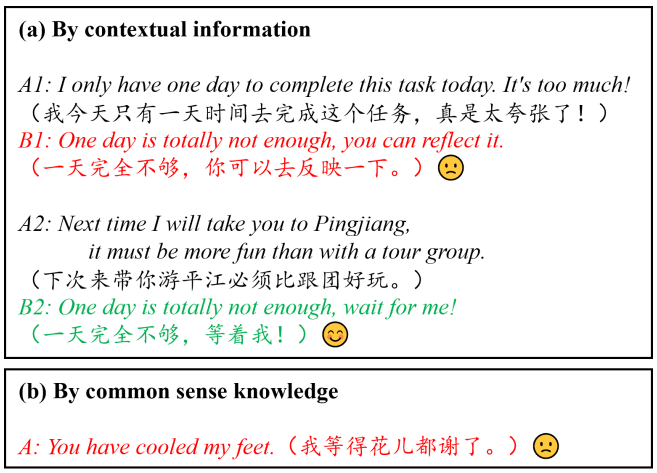


Figure 1: Sentences with different contexts or based on external common sense knowledge can express different sentiments. Expressions carrying positive (smiling face) or negative (crying face) sentiments are highlighted in color.

“implicit sentiment"[18]. For the task of text sentiment analysis in natural language processing, sentiment texts can be classified into explicit sentiment analysis and implicit sentiment analysis according to whether they contain obvious sentiment words in their expressions[15]. Statistical analysis of related studies[8, 5, 12] shows that implicit sentiment sentences are more common in Chinese, accounting for about 20% to 30% of all sentences with sentiment tendencies. Explicit sentiment sentences contain explicit sentiment vocabulary with strong semantic feature information, which can help sentiment analysis models accurately predict the sentiment tendency in sentences, and therefore many research results have emerged in the field of explicit sentiment analysis. On the contrary, implicit sentiment expressions do not have such characteristics, and the expressions are more obscure. For the current task of Chinese implicit sentiment analysis, there are two main critical problems faced:

Chinese implicit sentiment analysis requires the introduction of contextual information. Example (a) is an instance of the same expression expressing different sentimental tendencies in different contexts. The speaker conveys two opposite sentimental tendencies, negative (B1) and positive (B2), through the similar emotional expression “One day is totally not enough" according to the content received above. Thus, implicit sentiment analysis suffers from the problem that the sentiment expression itself does not provide enough information, and contextual information needs to be introduced.

Chinese implicit sentiment analysis lacks the support of external knowledge. Example (b) is a declarative sentence using metaphorical rhetoric. Although it does not contain any sentimental cues such as “anxious", etc., it effectively conveys the sentiment that the speaker has been waiting for a long time and is somewhat dissatisfied. Therefore, combined with human understanding of implicit sentimental expression, it is necessary to introduce an understanding of external common sense knowledge to support the discernment of implicit sentimental tendencies.

In order to address typical problems in Chinese implicit sentiment analysis tasks like those two listed above, we propose a sentiment analysis method in this paper that integrates contextual information and external common sense knowledge for implicit sentiment analysis.

The main contributions of this paper can be summarized as follows:

1. We propose a retrieval method to extract valid sentiment triads from a knowledge base and build an adaptive fusion of external common sense knowledge embedding layers to extend the semantic information of implicit sentiment expressions. Such improvements significantly facilitate the mining of implicit sentiment expressions hidden in ambiguous language fragments without sentiment words as clues.

2. Multipolar orthogonal attention mechanism is used to learn embedding representations for implicit sentiment expressions, and an attention layer that incorporates context is introduced to mine and combine valid information in the context. This effectively retains the differences between attention representations and thus contributes to distinguishing subtle differences between sentiment polarities.

3. The common sense knowledge information provided by the external knowledge base is fused with the contextual representation and the semantic information of the sentiment expression itself to achieve an effective extension of the implicit sentiment sentence representation and to perform sentiment label prediction. The experimental results show that our model obtains more satisfactory performance than the baseline models.

3 Related Work

Research related to implicit sentiment analysis is not fully developed compared to the field of explicit sentiment analysis[7, 15]. For implicit sentiment, it can be divided into factual implicit sentiment and rhetorical implicit sentiment. Rhetorical implicit sentiment can be subdivided into metaphorical or simile, rhetorical question, and ironic [12]. Early research on implicit sentiment analysis focused on the task of identifying metaphors [10], and as early as last century, Wilks et al. summarized metaphor, a common linguistic phenomenon, into a series of linguistic rules from the traditional cognitive and linguistic perspectives, through which metaphorical expressions and contexts, with the connections between them, are expressed. Building on this theory, Mason et al. [16] proposed a CorMet mapping of “source domain-target domain" that can be identified by a specific domain, based on the linguistic feature that words can express different meanings in different domains. The CorMet model, which identifies between specific domains, was proposed. Later, Lakoff [25] studied the process of metaphorical expression in depth and defined it as a conceptual mapping from the source domain to the target domain in terms of cognition, common sense, etc. In an earlier period, sentiment lexicon based methods played an important role in the field of sentiment analysis, but for implicit sentiment, the method could not be used directly for simple matching due to the lack of obvious sentiment words in the text variety. Shutova et al.[23] addressed this problem to some extent by using clustering to obtain the abstract metaphors embedded in words; based on this work, the researchers went further to study the distribution of metaphorical concepts by combining weakly supervised and unsupervised approaches [22], and after experimentally testing several hierarchical clustering methods under specific constraints, a metaphorical pattern recognition algorithm was proposed, which achieved good results on a multilingual test set. Zhang et al.[33] used sufficient collation and induction of sentiment texts to find that some phrases and special collocations, although without obvious sentiment cues, can implicitly express some implicit sentiment tendencies when they are expressed in sentences through metaphor or irony, and proposed a method to identify nouns with implicit sentiment domain features. Balahur et al.[1], on the other hand, focused on identifying the emotional tendency of the target with the help of contexts that do not contain emotion words, proposed an implicit sentiment analysis method based on common sense knowledge, and constructed an EmotiNet common sense knowledge base with relevant sentiment concepts; in subsequent work [2], the authors continued to improve their modeling method based on this common sense knowledge base and found that the sentiment of the target sentence could be discriminated from the context where all sentiments were implicitly expressed, and the experimental results showed that the method could effectively improve the effect of sentiment analysis. Zhao et al.[35] took an alternative approach by introducing pseudo-contextual information into implicit sentiment analysis, and obtained the best results at that time in comparison experiments. Tong et al. [26] developed a theory-based discriminant system to analyze the relevance of ambiguous expressions to influence the euphemism of implicit affective utterances. As deep learning methods are widely used in the field of sentiment analysis, many scholars have also started to use deep learning for implicit sentiment analysis tasks. Pulkit et al.[17] investigated recognition algorithms oriented to sarcastic expressions by using deep learning models. Qian et al.[19] applied

the idea of hierarchical classification to the task of identifying hate speech, and the method can effectively classify thirteen classes of hate speech categories.

In China, more and more scholars have started to focus on research targeting Chinese implicit sentiment analysis tasks in recent years. Liao et al.[13] first deconstructed the target text using syntactic analysis methods and proposed a representation learning framework based on tree convolution; immediately afterwards, the researchers in their paper [10, 12] provided a review of the work on Chinese implicit sentiment analysis, focusing on the characteristics of factual implicit sentiment; and based on word-level sentiment goals, sentence-level implicit sentiment expressions, and chapter-level contextual explicit sentiment tendencies, a multi-level semantic fusion representation learning approach was proposed for language modeling of factual implicit sentiment on two manually labeled factual implicit sentiment datasets. Experiments were conducted to verify that contextual information plays an important role in implicit sentiment recognition. Wei et al. [30] proposed a multi-polar orthogonal attention model MPOA by comparing the differences between affective tendency features in social media texts, which also introduced a BERT pre-trained language model to represent the texts, and the experimental results showed that the model can effectively bridge the differences between words and affective tendency polarity corresponding to attention. Zhao et al.[34] proposed a hybrid neural network model based on CNN-BiLSTM-Attention, which extracts text features by CNN, obtains contextual information by Bi-LSTM. In addition, the model enables the extraction of semantic and structural features from the word and sentence levels respectively, and highlights emotional information features that contribute more to emotional expressions by using the attention mechanism, and this hybrid model has improved the effectiveness of Chinese emotional utterance analysis. Zuo et al. [36] proposed a context-specific heterogeneous graph convolution model (CsHGCN) , which introduces the idea of heterogeneous graphs into implicit sentiment analysis tasks and builds a contextual representation framework through graph convolutional neural networks, with improved results compared to network approaches such as tree graph convolution and tree long and short-term memory networks. Yang et al. [31] proposed an implicit sentiment analysis model based on graph attention neural network, by constructing a heterogeneous graph of text and word correspondences, aggregating semantic information using graph convolutional networks, and calculating the contribution of words with to the sentiment expression of the text using an attention mechanism, thus allowing the model to focus on words that are more important for sentiment polarity. Chen et al.[6] proposed a parallel hybrid model of bidirectional long-short time neural network and context-aware tree recurrent neural network (CA-TRNN) for the problem of inaccurate feature information extraction in Chinese implicit sentiment analysis and gradient explosion or gradient disappearance for chapter-level text information extraction by existing serialization models, which effectively improved the accuracy of classification results with small time cost and better application capability. In addition to digging deeper into the semantic information in Chinese implicit sentiment texts, the recognition of implicit sentiment usually requires the introduction of external sentiment features and sentiment knowledge. Wang et al. [28] fused three levels of semantic information in sentiment expressions, and proposed a method based on hierarchical knowledge enhancement to alleviate the problem of "weak features" in Chinese implicit sentiment expressions, while introducing a multi-pooling approach to model the "multiple confusion weak features" problem. The accuracy of the model has been effectively improved, and the F1 score is 5.9% higher than the previous best model.

From the above review of existing research, it is clear that, in contrast to the task of explicit sentiment analysis, research on implicit sentiment analysis is still immature, remains in the exploratory stage and faces a large number of problems and challenges.

4 Methodology

4.1 Problem Setup

The definition of implicit sentiment in different research work [18, 20, 28] are slightly different. This paper integrates the expressions of the definitions, and defines implicit sentiment as: "a fragment of language that does

not directly contain obvious sentiment cues but expresses subjective sentiment", where “fragment of language" includes words, phrases and sentences, and does not include a paragraph composed of several sentences. On this basis, this paper defines implicit sentiment analysis task as the task of “sentiment analysis of Chinese implicit sentiment utterances that do not contain obvious sentiment words", and further abstracts it into a triple classification task, which classifies the possible sentiment tendencies of the sentences to be analyzed into ‘Positive’, ‘Neutral’, ‘Negative’.

The implicit sentiment analysis task is defined formally as follows: for any implicit sentiment sentence $s = \{w_1, w_2, \ldots, w_N\}$ and its context input to the model, output the model's prediction P_t of sentence s , and the Eq. 18 defines the Chinese implicit sentiment sentence classification task.

$$f(s) \rightarrow P_t = \{p_{t_0}, p_{t_1}, p_{t_2}\} \tag{18}$$

where N is the number of words contained in a given sentiment sentence, w_n is the nth word in the sentence, $n \in N$; p_t is the probability that the implicit sentiment sentence s belongs to a category, p_{t_0} denotes the probability that the implicit sentiment sentence is neutral, p_{t_1} denotes the probability that the implicit sentiment sentence is positive, p_{t_2} denotes the probability that the implicit sentiment sentence is negative.

4.2 Knowledge Graph And Knowledge Retrieval

Due to the lack of explicit sentiment words in implicit sentiment sentences, continuing to dig deeper into the semantic information latent in the text itself does not provide more adequate sentiment clues. At this point, it is possible to associate the process of solving the problem with the human brain, which is to draw on rich external background knowledge. Specifically, additional knowledge information can be introduced from the existing common sense knowledge graphs as sentiment cues for implicit sentiment analysis.

ConceptNet [1] is a comprehensive commonsense knowledge graph, which is essentially a large semantic network describing concepts such as words and phrases that people use and the commonsense relationships between them, covering spatial, physical, social, temporal, and psychological aspects of everyday life. ATOMIC [21] is an event-centric knowledge graph. Recently, based on ATOMIC, Li et al. [9] proposed a Chinese Commonsense Conversation Knowledge Graph (C3KG), which incorporates both social commonsense knowledge and dialog flow information, and is an important complement to Chinese common sense knowledge sources.

However, it is inefficient and unwise to retrieve sentiment knowledge triples directly in these huge knowledge bases which are oriented to almost all domains. Therefore, in this paper, a three-step triple extraction method is designed to extract valuable knowledge triples for the target implicit sentiment sentences.

Knowledge triples extraction based on rules The latest ConceptNet 5.5 includes a total of 34 different kinds of relations, and C3KG is divided into four main relationship patterns. The number of knowledge triples varies greatly across relationship types. After analyzing the knowledge triples in the Chinese part of ConceptNet5, we found that the top 5 types of relationships account for more than 60% of all relationships, including “Synonym", “RelatedTo", “Causes", “ExternalURL", and “MotivatedByGoal”. By observing the specific triple examples in ConceptNet5, it was obtained that the relations with the highest number of relations “Synonym", “RelatedTo" and others such as “AtLocation” have almost the same semantics between their corresponding head entity and tail entity or do not effectively provide additional sentiment cues, which leads to such relations being useless for sentiment analysis tasks. Instead, the model performance is greatly degraded by the large number of irrelevant knowledge retrieval processes and the introduction of redundant knowledge. Similarly, for C3KG, it is observed that “Emotion Cause Flow”, among the four relational models, has the greatest potential for implicit sentiment analysis tasks, while the other three relational models, “emotion Intention Flow”, “event flow”, and “concept flow”, can only extend semantic information at the factual level, but do not introduce sentiment cues well.

[1] https://conceptnet.io/

Therefore, for the knowledge triple $t_k = (h, r, t) \in T$ (where h and t represent the head entity and tail entity, respectively, and r denotes the relationship between two entities), the knowledge triples with these relationship types are firstly eliminated from the common knowledge base, which means only the triples with $r \in R$ are retained, and R denotes the set of valid relationships only after filtering.

Furthermore, in order to be able to introduce additional sentiment cues from the external knowledge graph efficiently, it is agreed that only one of the two, head entity h and tail entity t, exists in the target implicit sentiment sentence $s \in C$. With the above two rule constraints, it is possible to filter out those knowledge triples that introduce knowledge into the target implicit sentiment sentence.

Knowledge triples extraction based on sentiment dictionary matching After the rule-based knowledge triples extraction, in order to further extract the knowledge triples valuable for the implicit sentiment analysis task from the knowledge graph, explicit sentiment clues need to be introduced. For the knowledge triple $t_k = (h, r, t) \in T$, $h \in s$ or $t \in s$, the entity not in the target implicit sentiment sentence is matched using the sentiment dictionary d, which is the concatenation of the three mainstream Chinese sentiment dictionaries. Knowledge triples without explicit sentiment cue words were subsequently eliminated. After such matching and filtering, those sentiment knowledge triples with only one entity as an explicit sentiment word were selected to ensure that additional sentiment information could be introduced into the target implicit sentiment expression sentences.

Knowledge triples extraction based on event matching The sentiment knowledge triplets filtered by the above two filtering methods match the content of the target sentence at the character level. In order to be able to draw more fully on the relationship between events and sentiments embedded in the knowledge base, it is also necessary to extract sentiment events in the target sentences.

Sentiment expressions contain many colloquial expressions and complex sentence structures. In order to effectively extract sentiment events from the target implicit sentiment sentences, we adopt the same method for event extraction as that worked in the literature, and achieve effective extraction of sentiment events from the target implicit sentiment sentences through the methods and steps of Pre-processing, Verb-driven, Adjective-driven, and Recursive Applying. For example, for the implicit sentiment expression "Probably got a cold and stay up late, it is still some pain", the sentiment event "somebody gets cold" can be extracted, which corresponds to the C3KG and can be associated to the sentiment triple $(personX\ gets\ cold, xAttr, Uncomfortable)$, and "Uncomfortable" is a negative sentiment tendency in the sentiment dictionary, thus introducing explicit sentiment cues.

Finally, since ConceptNet and C3KG are multilingual general knowledge base, although it contains some knowledge triples in Chinese, it is still more based on entity relationship pairs in English or other languages. For this reason, the knowledge in those knowledge bases are translated into Chinese as well as aligned and merged with the Chinese part through online translation. The merged knowledge graph is finally used as the main source of external knowledge in this paper.

4.3 Sentiment Analysis Incorporating Contextual Information And External Common Sense Knowledge

This method adaptively fuses knowledge from external common sense knowledge graphs by designing knowledge retrieval mechanisms in this paper in the process of learning textual representations. By doing so, a process similar to that of the human brain in understanding implicit emotional expressions is simulated, i.e., drawing on one's own experience or common sense to distinguish sentiment tendencies, allowing the model to achieve a deeper understanding of the text beyond the literal semantic information.

The above model network framework diagram is shown in Fig 2 . The model consists of four layers: input layer, adaptive knowledge fusion layer, multipolar attention layer fusing contextual contexts, and output layer. The structure of each layer is described in detail in the following.

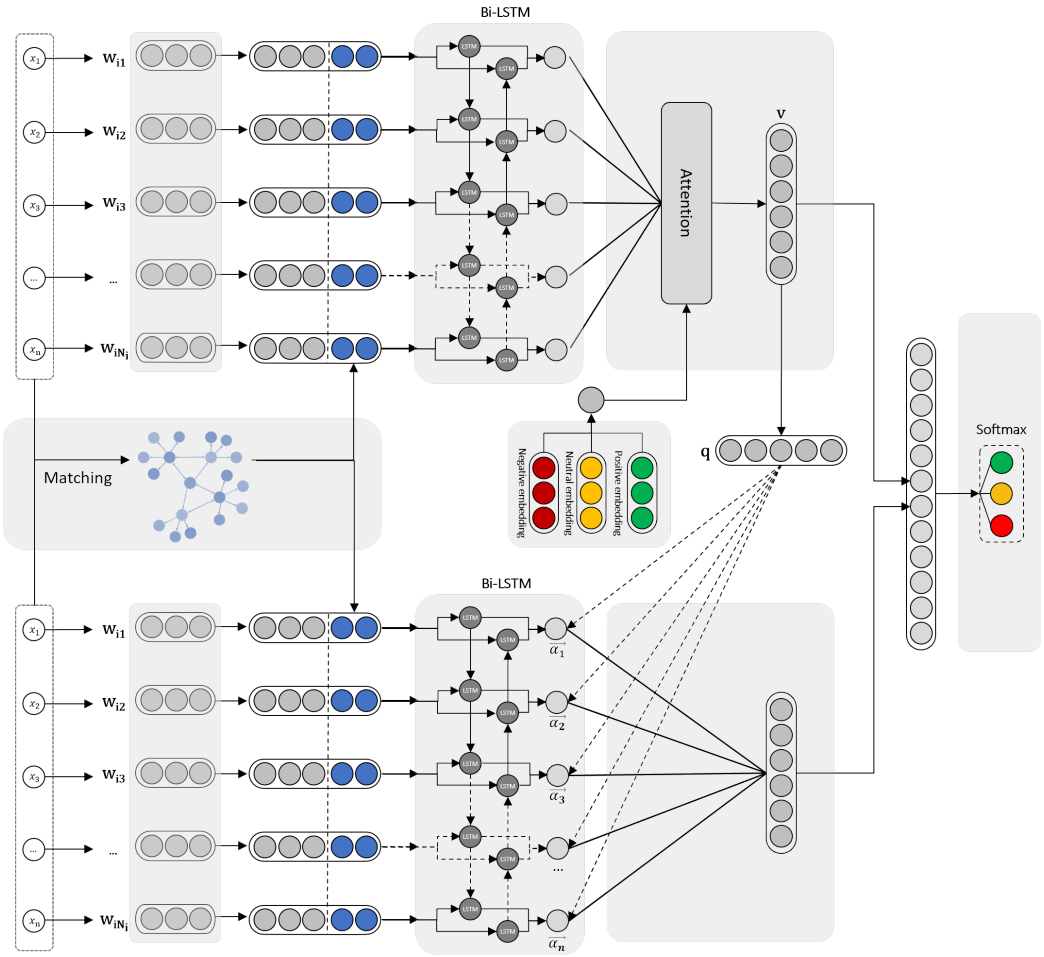


Figure 2: Framework of the model

Input layer For the input of the pre-trained model, the implicit sentiment text needs to be encoded first. For the original corpus input, each word or phrase x_i in the sentence is hard-coded into the corresponding vector representation by one-hot encoding representation. The pretrained model uses Transformer as its feature extractor, and also needs to encode the word-by-word position information in the text to obtain the word position representation vector. Finally, the above vector representations are combined and input to the pretrained model to obtain the continuous word embeddings $\vec{w}_t \in R^{d_e}$ encoded with contextual information, which are used as input information for the next layer.

Adaptive knowledge fusion layer In order to better fuse the extracted sentiment knowledge triples with the target implicit sentiment sentence information, a knowledge fusion strategy is needed. For each target implicit sentiment sentence s, more than one sentiment knowledge triple may be extracted, i.e., the set of triplets $G = \{g_1, g_2, \ldots, g_n\}$ can be obtained. The set of knowledge triplets G extracted from it is regarded as a knowledge graph $G^i = \{g_1^i, g_2^i, \ldots, g_n^i\}$. Using the TranR model [14] for each knowledge triple $g_j^i = (h_j, r_j, t_j)$ in G^i, we obtain the knowledge embedding representation $\boldsymbol{g}_j^i = (\boldsymbol{h}_j, \boldsymbol{r}_j, \boldsymbol{t}_j)$. The head entity h is designated as the entity in the target implicit sentiment sentence, and the tail entity t is the entity in the external sentiment knowledge graph. Using the representations of all head and tail entities learned in the knowledge graph G^i in the knowledge subgraphs, their weights are calculated adaptively by the attention mechanism, and the embedded representations of the knowledge subgraphs are learned by this to represent the sentiment cues introduced for the

target implicit sentiment sentence, calculated as shown in the Equ. (19).

$$\boldsymbol{G}^i = \sum_{j=1}^{n}(\text{softmax}(\boldsymbol{W}_{tv}\boldsymbol{t}_j\frac{\boldsymbol{t}_j^T\boldsymbol{W}_{tk}^T\boldsymbol{W}_h\boldsymbol{h}_j}{\sqrt{d}}) + \boldsymbol{W}_r\boldsymbol{r}_j) \tag{19}$$

where $\boldsymbol{W}_h, \boldsymbol{W}_{tk}, \boldsymbol{W}_{tv}, \boldsymbol{W}_r \in \mathbb{R}^{l\times m}$ are the parameter matrices of head entities, tail entities and relations. $\boldsymbol{h}_j, \boldsymbol{t}_j, \boldsymbol{r}_j \in \mathbb{R}^m$.

The knowledge graph embedding representation $\boldsymbol{G}^i$ obtained by learning represents the additional external common knowledge introduced, and if the target sentiment sentence cannot be matched to any sentiment knowledge triplet, $\boldsymbol{G}^i$ is treated as a zero vector. The word embedding $\boldsymbol{w}_i$ obtained in the "input layer" is fused with $\boldsymbol{G}^i$ to achieve the introduction of external common sense knowledge, and finally the result $\boldsymbol{e}_i = \boldsymbol{w}_i \oplus \boldsymbol{G}^i$ is input to the next layer.

Multipolar attention layer fusing contextual contexts This layer models the fusion of implicit sentiment expressions and contextual information using multipolar attention mechanisms to extend the information in implicit sentiment expressions.

As for the text implicit sentiment classification task, the output of the current moment is not only related to the state before the current moment, but may also related to the state after the current moment. In this layer, the input of the upper layer network first passes through a bi-directional long and short-term memory network (Bi-LSTM), Bi-LSTM is composed of an LSTM with forward processing sequences and an LSTM with reverse processing sequences, considering both forward and backward sequences, making full use of contextual information and allowing deep feature extraction of the contextual information inside the input sentence.

As mentioned in the literature [30], the difference between attention weights for specific polarities is an important feature of the Chinese implicit sentiment analysis task. Therefore, multi-polar attention mechanisms are introduced in this layer to incorporate sentiment polarity labeling information into the implicit sentiment representation. Specifically, polarity-specific queries are introduced by using attention mechanisms with different polarities in order to make the attention mechanism more focused on the differences between different sentiment polarities.The calculation formula are shown in Eq. (20), Eq. (21), Eq. (22).

$$\boldsymbol{v}^q = \sum_{i=1}^{T}\alpha_i^q\boldsymbol{t}_i \tag{20}$$

$$\alpha_i^q = \frac{\exp\left(e_i^q\right)}{\sum_{k=1}^{T}\exp\left(e_k^q\right)} \tag{21}$$

$$e_i^q = \boldsymbol{q}\boldsymbol{W}\boldsymbol{m}_i \tag{22}$$

where $\boldsymbol{v}^q$ represents the embedding representation of sentences under sentiment polarity q. T is the length of the target implicit sentiment sentence.α_i is the normalized weight of the word or character w_i computed from the Bi-LSTM output $\boldsymbol{t}_i$. e_i^q is the attention score of w_i. The matrix $\boldsymbol{W}$ is then used to query the weight matrix of q and $\boldsymbol{h}_i$ in the bilinear function. $\boldsymbol{q} \in \boldsymbol{Q} = \left\{\boldsymbol{q}_{\text{pos}}, \boldsymbol{q}_{\text{neg}}, \boldsymbol{q}_{\text{neu}}\right\}$ correspond to polarity-specific query embedding representations of positive, negative and neutral sentiment polarity, respectively. The same sentiment dictionary as in the knowledge triad extraction process above is used here to initialize the polarity query embedding representation, which is computed by averaging the embeddings of sentiment words with the same sentiment polarity in the sentiment dictionary, as shown in Eq. (23):

$$\boldsymbol{q} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{w}_i^q \tag{23}$$

$\boldsymbol{w}_i^q$ is a pre-trained embedding representation of the word or character w_i whose sentiment polarity belongs to q, $q \in Q = \{q_{\text{pos}}, q_{\text{neg}}, q_{\text{neu}}\}$.

The implicit sentiment sentence representations with weights under the three sentiment polarities are spliced to obtain a new representation of the implicit sentiment expression $\boldsymbol{v} = \boldsymbol{v}^{q_{pos}} \oplus \boldsymbol{v}^{q_{neg}} \oplus \boldsymbol{v}^{q_{neu}}$.

The context representation is considered as a whole, and the implicit sentiment sentence representation q^c transformed by the fully connected layer is used as the query vector to assign weights to each word in C in order to achieve the extraction of key features in the contextual information of the target implicit sentiment expression by the model through the attention mechanism.

$$\boldsymbol{v^c} = \sum_{i=1}^{T_C} \alpha_j^c \boldsymbol{t}_i \tag{24}$$

$$\alpha_j^c = \frac{\exp\left(e_j^c\right)}{\sum_{k=1}^{m} \exp\left(e_k^c\right)}, \tag{25}$$

$$e_j^c = \boldsymbol{q}^c \boldsymbol{W}^c \boldsymbol{h}_j^c \tag{26}$$

$$\boldsymbol{q}^c = \tanh(\boldsymbol{W}\boldsymbol{v} + \boldsymbol{b}) \tag{27}$$

Finally, by splicing the implicit sentiment sentence representation $\boldsymbol{v}$ with the contextual representation $\boldsymbol{v^c}$, the implicit sentiment sentence representation $\boldsymbol{v_{out}}$, which incorporates the contextual information, is obtained as the output of this layer, and it is fed into the output layer for the final probability calculation, as shown in Equ. (28):

$$\boldsymbol{v_{out}} = \boldsymbol{v} \oplus \boldsymbol{v^c} \tag{28}$$

Output layer In the output layer, an implicit sentiment sentence representation $\boldsymbol{v_{out}}$ incorporating contextual information is projected into the sentiment polarity category space using a fully connected network. Then, the scores of each category are normalized to an approximate probability value $\widehat{\boldsymbol{y}}$ by the calculation of the Eq. (29).

$$\widehat{\boldsymbol{y}} = \text{softmax}\left(\boldsymbol{W}_{\text{out}}\, \boldsymbol{v}^{\text{T}} + \boldsymbol{b}_{\text{out}}\right) \tag{29}$$

where $\boldsymbol{W}_{\text{out}}$ and $\boldsymbol{b}_{\text{out}}$ are the weight matrix and bias, respectively.

5 Experimental Setup

In this section, we present the details of the datasets used, the methods, and the implementation details of our models.

5.1 Dataset

In the field of Chinese implicit sentiment analysis research, the most widely used public dataset is the dataset used in the SMP2019 Chinese Implicit Sentiment Analysis Review (SMP-ECISA 2019) provided by Shanxi University [2]. The corpus of this dataset is collected from social media such as microblogs, travel websites, and product e-commerce, and a large-scale sentiment dictionary is used to filter the collected data, remove sentiment expression sentences with explicit sentiment words, and use the remaining text data as the Chinese implicit sentiment analysis dataset. The main domains/topics covered in this dataset include but are not limited to: Spring

[2]https://www.biendata.xyz/competition/ smpecisa2019/data/

Table 13: SMP-ECISA 2019 Dataset

	articles	labeled sentences	neutral	positive	negative
Training set	12,664	1,4774	6,989	3,828	3,957
Validation set	4,391	5,143	2,553	1,232	1,358
Test set	6,380	/	/	/	/
After merging	17,055	19,917	9,542	5,060	5,315

Festival Gala, Haze, LeTV, National Exam, Travel, Dragon Boat Festival, etc. The classification of sentiment labels in the dataset is divided into 3 types: negative, neutral and positive, corresponding to the numerical labels 1, 2 and 0. Tab. 13 shows the detailed statistics of the dataset.

In order to evaluate and compare existing methods and our method more fairly, we combined the training and validation sets to obtain 17,055 articles with a total of 19,917 labeled data, and re-divided the training set, validation set, and test set in the ratio of 8:1:1, and evaluated them using ten-fold cross-validation, and took the best result as the final result of the model.

5.2 Evaluation Metrics

To more reasonably and reliably measure the effectiveness of the method proposed in this paper, the F1 scores of the three emotional polarities and the macro-average F1 scores are calculated as indicators in this paper, as shown in the Eq. (30), Eq. (31).

$$F_1 i = \frac{2 * P_i * R_i}{P_i + R_i}, \tag{30}$$

$$F_1 - \text{macro} = \frac{1}{N} \sum_{i=1}^{N} F_1 i, \tag{31}$$

$i \in \{$ positive, negative, neutral $\}$ is one of the three sentiment polarities, and P_i and R_i are the accuracy and recall corresponding to the sentiment polarity i.

5.3 Implementation Details

Due to the limitation of the training hardware equipment, we can not train all the corpus at the same time, so the complete training set needs to be divided into many training batches. Based on a careful observation of the dataset, it was found that the corpus data in the dataset varied in length, which led to the internal setting of the model for the input length that may affect the final results of the model. Taking into account the hardware resources of the training environment, we chose 128 as the maximum length of input sentences, with each batch size of 16, and trained 5 epochs in total. For each single item in the BiLSTM, each has a hidden layer unit that needs to be learned by setting different sizes that will change the model's ability to capture features, and we set them all to 128. Additionally, the learning rate was set to 5e-5 and the dropout rate was 0.4.

5.4 Baseline Methods

Because of the limited research on the implicit sentiment analysis task of Chinese text, in this section, in addition to selecting some of the latest implicit sentiment analysis models according to the settings in the literature [11], some representative models with good results in traditional explicit sentiment analysis tasks[32] are selected as baselines for comparison experiments. The baseline experiments are described in detail as follows.

Bi-LSTM+multi-attention The combination of Bi-LSTM combined with attention mechanism has been proven to be an effective model in many tasks related to sentiment analysis. The multi-headed attention mechanism can capture more of the differences between multiple attentions to some extent and enhance the effect even further.

Bi-LSTM+MPOA Jiyao et al. [30] proposed an orthogonal multi-attention mechanism and fused it with Bi-LSTM to improve the traditional multi-attention mechanism by using orthogonal attention with specific sentiment polarity.

CMPOA The model that Wang et al. [29] proposed integrates contextual information into the basic MPOA to compensate for the lack of information in the implicit sentiment sentence itself.

CsHGCN Zuo et al. [36] proposed a heterogeneous graph convolutional network model to represent the implicit sentiment sentences. The entire context is treated as a heterogeneous graph at the document level and deep domain features are captured by graph convolution.

Tree-LSTM+KG Syntactic structure information is important for modeling implicit sentiment sentences. The syntactic structure tree of the input sentence is obtained using LTP [4], and then a Tree-LSTM [24] based model is constructed to fuse the syntactic structure to the embedding of the sentence.

6 Results And Analysis

6.1 Comparison with Baselines

Experiment results are reported in Table. 14, and by observing and analyzing the results it can be found that:

From the overall view of each metric, the method proposed in this paper has the best results on all metrics, indicating that our method can effectively improve the effectiveness of the implicit sentiment analysis task. Specifically, in our experiments, we compare the model proposed in this paper with models that have been effective in traditional explicit sentiment analysis tasks and with existing implicit sentiment analysis models, and our model obtained the best results.

Our model has a large improvement in the implicit sentiment analysis task compared to the traditional explicit sentiment analysis model, which indicates that the traditional explicit sentiment analysis approach has some shortcomings in the implicit sentiment analysis task. Further, from the results of the model without the introduction of common sense knowledge information, it can be found that the introduction of additional external common sense knowledge has a 4% improvement in the $F_1 - macro$ metric for the implicit sentiment analysis task, which indicates that the introduction of the triplet of common sense knowledge can provide critical sentiment information and can effectively compensate for the lack of information caused by the absence of explicit sentiment words. On the other hand, the result analysis of the model without introducing contextual information illustrates that for the target implicit sentiment expression, fusion with contextual information can extend the sentiment semantic information of implicit sentiment sentences from another aspect to enhance the implicit sentiment analysis. The above results and analysis show that effective extension of the implicit sentiment sentence representation can be achieved by fusing the common sense knowledge information provided by the external knowledge base with the contextual representation and the semantic information of the sentiment expression itself.

Meanwhile, the common sense knowledge fusion method proposed in this paper can effectively improve the interpretability of the model. Traditional methods, such as BiLSTM+MPOA, usually interpret the prediction results based on the attention weights of words in the sentence only. However, in implicit sentiment analysis tasks, words with higher attention weights are usually also neutral or objective words. The reason for the model

Table 14: Performance of Chinese implicit sentiment analysis

Methods	**PLM**	F_{neu}	F_{pos}	F_{neg}	F_1**-macro**
SDT-CNN		0.735	0.701	0.697	0.711
TextCNN		0.734	0.724	0.712	0.723
CsHGCN		0.848	0.670	0.775	0.764
BiLSTM + Att		0.800	0.630	0.747	0.726
BiLSTM + Mult-Att		0.810	0.634	0.749	0.730
	random	0.800	0.568	0.702	0.690
CMPOA	BERT	0.784	0.585	0.712	0.694
	ERNIE	0.816	0.662	0.761	0.746
	random	0.750	0.560	0.692	0.667
TreeLSTM+KG	BERT	0.764	0.581	0.725	0.690
	ERNIE	0.805	0.650	0.779	0.745
	random	0.764	0.569	0.691	0.675
BiLSTM+MPOA	BERT	0.808	0.684	0.669	0.720
	ERNIE	0.834	0.690	0.791	0.775
	random	0.759	0.558	0.690	0.669
LSTM+Att+KG	BERT	0.771	0.593	0.732	0.699
	ERNIE	0.802	0.643	0.781	0.742
	random	0.773	0.554	0.692	0.673
BiLSTM+MPOA+KG	BERT	0.792	0.593	0.710	0.699
	ERNIE	0.819	0.655	0.749	0.741
	random	0.789	0.571	0.702	0.687
Ours	BERT	0.803	0.582	0.716	0.699
	ERNIE	**0.849**	**0.690**	**0.799**	**0.778**

prediction can be demonstrated by introducing external common sense knowledge. Take the sentence “I got a good grade in this exam" as an example, the positive sentiment predicted by the model can be explained by the sentiment triple $(getting\ a\ good\ grade, causes, praise)$, which makes the prediction more intuitive and convincing. In summary, the experimental results show that the method in this paper has improved the performance of Chinese implicit sentiment analysis and has shown the ability to interpret the results to a certain extent.

Table 15: Performance of Chinese implicit sentiment analysis The ablations of different components are reported separately in this table, where the model without additional explicit introduction of external common sense knowledge is denoted as ‘-KG’, ‘-MPA’ indicates that no multipolar orthogonal attention mechanism is used in the process of sentiment representation and knowledge incorporation, and ‘-Context’ indicates that no additional contextual information of additional sentiment representation is introduced.

Methods	F_{neu}	F_{pos}	F_{neg}	F_1**-macro**
Ours	**0.849**	**0.690**	**0.799**	**0.778**
-KG	0.816	0.662	0.761	0.746
-MPA	0.827	0.674	0.782	0.761
-Context	0.819	0.655	0.749	0.741

6.2 Ablation Study

Tab. 15 shows the results of the metrics after removing each component from our model separately. The results of the three ablation experiments show that each component of the proposed method in this paper contributes to the model to some extent.

Specifically, removing the introduction of external knowledge has the greatest impact on the model's effectivenes; without the use of external common sense knowledge ('-KG'), we observe a more dramatic decrease in performance compared to all other components, with a decrease of nearly 4.1% in $F_1 - macro$, demonstrating the effectiveness and importance of introducing additional common sense knowledge for the implicit sentiment analysis task, which can effectively complement the implicit sentiment expression sentiment cues that are missing due to the absence of explicit sentiment vocabulary. The next most effective effect of removing the introduction of contextual information on the model, with an overall performance decrease of 3.4% when focusing only on the semantic information of the implicit sentiment expression itself (-Context'), which indicates that contextual information is also more significant for understanding the implicit sentiment expression and can be used when the implicit sentiment expression. This indicates that contextual information is also important for understanding implicit sentiment expressions, and can offer effective sentiment semantic information when the implicit sentiment expressions cannot provide sufficient sentiment features by their own information. Finally, the multi-polar attention mechanism has the least effect on the model, and the addition of MPA mechanism improves the model by about 2.1%, which also indicates that the MPA mechanism is somewhat helpful, but not as effective as the above two components.

In summary, the various components used in the model proposed in this paper are all instrumental for the implicit sentiment analysis task.

7 Conclusion

In this paper, we propose a sentiment analysis method incorporating contextual information and external common sense knowledge for implicit sentiment analysis. We first propose a retrieval method to extract valid sentiment triads from a knowledge base and build an adaptive fusion of external common sense knowledge embedding layers to extend the semantic information of implicit sentiment expressions. Multi-polar attention mechanism is then used to learn embedding representations for implicit sentiment expressions, and an attention layer that incorporates context is introduced to mine and combine valid information in the context. Finally, the common sense knowledge information provided by the external knowledge base is fused with the contextual representation and the semantic information of the sentiment expression itself to achieve an effective extension of the implicit sentiment sentence representation and to perform sentiment label prediction. Experimental results on the SMP2019-ECISA dataset show that the method outperforms existing state-of-the-art methods and can fully exploit contextual information and external common sense knowledge to effectively improve the analysis of implicit sentiment expressions.

8 Acknowledgments

This work was supported by National Natural Science Foundation of China (No.62176185)

References

[1] Alexandra Balahur, Jesús M Hermida, and Andrés Montoyo. Detecting implicit expressions of sentiment in text based on commonsense knowledge. In Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011), pages 53–60, 2011.

[2] Alexandra Balahur, Jesús M Hermida, and Andrés Montoyo. Detecting implicit expressions of emotion in text: A comparative analysis. Decision support systems, 53(4):742–753, 2012.

[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] Wanxiang Che, Zhenghua Li, and Ting Liu. Ltp: A chinese language technology platform. In Coling 2010: Demonstrations, pages 13–16, 2010.

[5] Huan-Yuan Chen and Hsin-Hsi Chen. Implicit polarity and implicit aspect recognition in opinion mining. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 20–25, 2016.

[6] Qiuchang CHEN, Hui ZHAO, Enguang ZUO, ZHAO Yuxia, and Wenyu WEI. Implicit sentiment analysis based on context aware tree recurrent neutral network. Computer Engineering and Applications, 58(07):167–175.

[7] Cambria Erik, Poria Soujanya, Gelbukh Alexander, and Thelwall Mike. Sentiment analysis is a big suitcase. IEEE Intelligent Systems, 32(6):74–80, 2017.

[8] Liao Jian, Li Yang, and Wang Suge. The constitution of a fine-grained opinion annotated corpus on weibo. In Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data, pages 227–240. Springer, 2016.

[9] Dawei Li, Yanran Li, Jiayi Zhang, Ke Li, Chen Wei, Jianwei Cui, and Bin Wang. C3kg: A chinese commonsense conversation knowledge graph. arXiv preprint arXiv:2204.02549, 2022.

[10] Jian Liao. Research on Fact-implied Implicit Sentiment Analysis Based on Representation Learning. PhD thesis, Shanxi University, 2018.

[11] Jian Liao, Min Wang, Xin Chen, Suge Wang, and Kai Zhang. Dynamic commonsense knowledge fused method for chinese implicit sentiment analysis. Information Processing & Management, 59(3):102934, 2022.

[12] Jian Liao, Suge Wang, and Deyu Li. Identification of fact-implied implicit sentiment based on multi-level semantic fused representation. Knowledge-Based Systems, 165:197–207, 2019.

[13] Jian Liao, Suge Wang, Deyu Li, and Xiaoli Li. Freerl: Fusion relation embedded representation learning framework for aspect extraction. Knowledge-Based Systems, 135:9–17, 2017.

[14] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In Twenty-ninth AAAI conference on artificial intelligence, 2015.

[15] Bing Liu. Sentiment analysis: Mining opinions, sentiments, and emotions. Cambridge university press, 2020.

[16] Zachary J Mason. Cormet: a computational, corpus-based conventional metaphor extraction system. Computational linguistics, 30(1):23–44, 2004.

[17] Pulkit Mehndiratta, Shelly Sachdeva, and Devpriya Soni. Detection of sarcasm in text data using deep convolutional neural networks. Scalable Computing: Practice and Experience, 18(3):219–228, 2017.

[18] Donghang PAN, Jingling YUAN, Lin Li, and Deming SHENG. A chinese implicit sentiment classification model combining contextual features. Computer Engineering & Science, 42(2):10, 2020.

[19] Jing Qian, Mai ElSherief, Elizabeth Belding, and William Yang Wang. Hierarchical cvae for fine-grained hate speech classification. arXiv preprint arXiv:1809.00088, 2018.

[20] David Samuel, Jeremy Barnes, Robin Kurtz, Stephan Oepen, Lilja Øvrelid, and Erik Velldal. Direct parsing to sentiment graphs. arXiv preprint arXiv:2203.13209, 2022.

[21] Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. Atomic: An atlas of machine commonsense for if-then reasoning. In Proceedings of the AAAI conference on artificial intelligence, volume 33, pages 3027–3035, 2019.

[22] Ekaterina Shutova, Lin Sun, Elkin Darío Gutiérrez, Patricia Lichtenstein, and Srini Narayanan. Multilingual metaphor processing: Experiments with semi-supervised and unsupervised learning. Computational Linguistics, 43(1):71–123, 2017.

[23] Ekaterina Shutova, Simone Teufel, and Anna Korhonen. Statistical metaphor processing. Computational Linguistics, 39(2):301–353, 2013.

[24] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075, 2015.

[25] J. P. Thorne. Lakoffgeorge and johnsonmark, metaphors we live by. chicago and london: The university of chicago press, 1980. pp. xiii + 242. bolingerdwight, language the loaded weapon: the use & abuse of language today. london and new york: Longman, 1980. pp. ix + 214. Journal of Linguistics, 19(1):245–248, 2008.

[26] Eddie MW Tong, Deborah H Tan, and Yan Lin Tan. Can implicit appraisal concepts produce emotion-specific effects? a focus on unfairness and anger. Consciousness and cognition, 22(2):449–460, 2013.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[28] Hongbin Wang, Minghui Hou, Fan Li, and Yafei Zhang. Chinese implicit sentiment analysis based on hierarchical knowledge enhancement and multi-pooling. IEEE Access, 8:126051–126065, 2020.

[29] Suge WANG, Min WANG, Jian LIAO, and Xin CHEN. An implicit sentiment sentence identification method base on context information. Journal of Shanxi University(Natural Science Edition), 45(02):380–386.

[30] Jiyao Wei, Jian Liao, Zhenfei Yang, Suge Wang, and Qiang Zhao. Bi-lstm with multi-polarity orthogonal attention for implicit sentiment analysis. Neurocomputing, 383:165–173, 2020.

[31] Shanliang YANG and CHANG Zheng. Chinese implicit sentiment analysis based on graph attention neural network. Computer Engineering and Applications, 57(24):161–167.

[32] Kai Zhang, Kun Zhang, Mengdi Zhang, Hongke Zhao, Qi Liu, Wei Wu, and Enhong Chen. Incorporating dynamic semantics into pre-trained language model for aspect-based sentiment analysis. arXiv preprint arXiv:2203.16369, 2022.

[33] Lei Zhang and Bing Liu. Identifying noun product features that imply opinions. In Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies, pages 575–580, 2011.

[34] Rongmei ZHAO, Xi XIONG, Shenggen JU, Zhongzhi LI, and Chuan XIE. Implicit sentiment analysis for chinese texts based on a hybrid neural network. Journal of Sichuan University(Natural Science Edition), 57(2):264–270, 2020.

[35] Yanyan Zhao, Bing Qin, and Ting Liu. Collocation polarity disambiguation using web-based pseudo contexts. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 160–170, 2012.

[36] Enguang Zuo, Hui Zhao, Bo Chen, and Qiuchang Chen. Context-specific heterogeneous graph convolutional network for implicit sentiment analysis. IEEE Access, 8:37967–37975, 2020.

Hypergraph Clustering Network for Interaction Data

Tianchi Yang[†] Luhao Zhang[‡] Cheng Yang[†]
Chuan Shi[†] Maodi Hu[†] Tao Li[†] Dong Wang[†]
[†] Beijing University of Posts and Telecommunications, Beijing, China
{yangtianchi,shichuan}@bupt.edu.cn, albertyang33@gmail.com
[‡] Meituan, Beijing, China
{zhangluhao,litao19,wangdong07}@meituan.com, Londeehu@gmail.com

Abstract

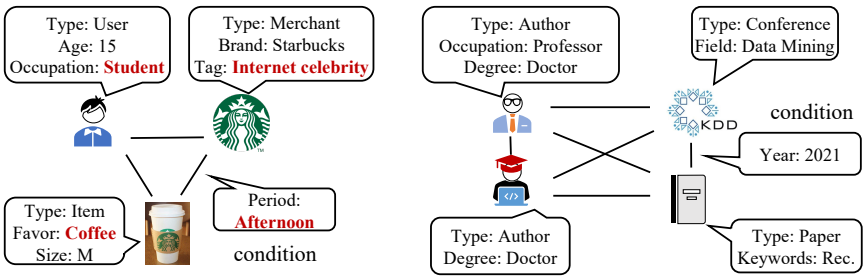
With the rapid development of social media, the data generated from interaction, which is an action collaboratively done by several objects under a certain condition, have grown exponentially. It triggers an urgent need to cluster interaction data for implicit patterns benefiting downstream tasks. Although clustering methods have been extensively studied for a long time, they mainly focus on node/object clustering, which breaks the intrinsic collaborative associations among the involved objects and conditions, thus limiting clustering performance of interaction data. To tackle this issue, we propose a novel Hypergraph CLustering network for Interaction Data (HyCLID), which not only clusters whole interactions rather than individual objects in interactions, but also exploits correlation among attributes of objects and conditions. Specifically, we first construct an attributed hypergraph to model interactions. Then, we propose a novel rethinking-based hypergraph neural network to learn the representation of interactions, which employs a novel attentive routing-based rethinking mechanism to capture the correlations among multiple object attributes and condition attributes. Furthermore, a novel adaptive mini-batch method is designed for the large scale of interaction data. Experimental results demonstrate the effectiveness of our methods for clustering the interactions and the practical value of the discovered interaction patterns.

1 Introduction

With the rapid development of social media, the volume of interaction data[1] has grown exponentially to an overwhelming scale [13, 30]. Generally, an interaction is an action collaboratively done by several objects under a certain condition [30]. For example, as illustrated in Figure 1, "a user ordered a cup of latte at Starbucks in the afternoon" in food delivery systems, and "two researchers collaborated to publish a paper in WSDM in 2022" in academic networks, etc. In detail, Interaction 1 involves a condition about period and three objects including a user, a merchant and a product, each of which carries several attributes, which indicates the characteristic of interaction data is that the interaction involves multiple objects and the rich correlations among the attributes of objects and conditions. As interaction data play an increasingly important role in daily life, analyzing them becomes a priority.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1]The term interaction data is similar to the term relational data, which focuses on the data that there are pair-wise relations *between* objects, while we introduce the term interaction data in this paper for distinction to focus on the relations among multiple objects.



Interaction 1: a user ordered a cup of latte at Starbucks in the afternoon

Interaction 2: two researchers collaborated to publish a paper in KDD in 2021

Figure 1: Examples of interactions from Meituan Waimai[2] and academic networks.

As a fundamental task in data mining, clustering on interaction data can discover the underlying patterns beneficial to downstream tasks. For instance, in the above example, the user preference can be found by clustering his interactions, enabling the platform to provide better recommendation services. Traditional clustering focuses on object features, such as K-means [6, 32], etc. Besides, graph-based clustering, which treats objects as nodes in a graph and leverages structural information of objects for clustering the objects, has also been widely explored, e.g., spectral clustering [16]. Recently, some researchers pay attention to attributed graph clustering, which jointly considers node attributes and graph structure [28]. A noticeable trend is the wide adoption of deep learning in clustering [15]. For example, Graph Neural Network (GNN) based clustering methods [1, 21, 18, 29] can effectively learn object representations for clustering via message passing mechanism. However, we argue that existing methods could not fully cope with the interaction data in the real world. Specifically, in interaction data, multiple objects as well as interaction conditions collaboratively form an indivisible semantic unit. Nonetheless, these methods focus on clustering individual nodes/objects, which breaks the intrinsic collaborative associations among the involved objects and conditions, thus limiting their ability to discover the underlying interaction patterns. Take Interaction 1 in Figure 1 as an example, if we ignore the associations among user, merchant and product, one may discover a one-sided pattern that "this user loves coffee". This pattern will possibly lead to an inappropriate recommendation of high-end coffee to this student far beyond his consumption level.

Therefore, we propose to cluster interactions instead of objects, where each interaction is regarded as a basic unit to be clustered. Nevertheless, this is not a trivial task due to the following reasons: (1) *How to effectively model interaction data involving multiple objects?* As mentioned before, an interaction is an indivisible semantic unit. Nevertheless, a majority of existing solutions employ graph structure to model the relations between objects, which will break the collaborative association among multiple objects into several pairwise sub-relations. Hence they are insufficient to model interaction data in the real world. (2) *How to capture the correlations among the attributes of objects and conditions?* The occurrence of an interaction is mainly owing to the complex correlation among rich attributes of its involved objects and conditions. For instance, Interaction 1 in Figure 1 is mainly derived by the correlation among "student, Internet celebrity, coffee, afternoon" (marked in red). Since this correlation is among multiple attributes, capturing them is a combinatorial optimization problem. To avoid combinatorial explosion, some recent methods [19, 24] adopt self-attentive layers to capture the pairwise correlations between two features, called second-order features. Then, the higher-order features can be approximated by stacking multiple self-attentive layers with residual connections. However, the errors in the bottom layers will inevitably propagate to the following layers. Even worse, such error propagation will be enlarged as the number of layers increases, thus possibly leading to the omissions of some potential high-order features. (3) *How to handle the large scale of interaction data during clustering?* The scale of interaction data in real applications is always large, thus requiring clustering in a batch fashion. Nonetheless, the data distributions

[2]Meituan Waimai is a food delivery platform. https://waimai.meituan.com

of different batches are usually not the same, which probably causes data imbalance so that the clustering error may largely fluctuate or even increase [26]. Unfortunately, existing methods [23, 21] cannot automatically make adaptive adjustments for different batches, thereby leading to sub-optimal clustering performance.

To tackle the aforementioned issues, we propose a novel Hypergraph CLustering network for Interaction Data (HyCLID). Specifically, to model interactions that involve multiple objects and conditions, we construct an attributed hypergraph for interactions. As the hyperedges can connect an arbitrary number of objects, we model each interaction as a hyperedge connecting several nodes that represent its involved objects. Then, we propose a novel rethinking-based hypergraph neural network to capture the complex correlations (high-order features) among object attributes and conditions for embedding the interactions (hyperedges). Besides, to avoid omissions of potential high-order features, we design a novel attentive routing-based rethinking mechanism to review the capturing process and correct the errors in the bottom layers. Finally, we propose an adaptive mini-batch clustering method based on the learned interaction representations to perform deep clustering on large-scale interaction data, which employs an adaptive batch standardization to remove the impact of different data distributions of different batches. The main contributions are summarized as follows:

- We propose a hypergraph clustering network for interaction data, namely HyCLID. With interactions modeled as an attributed hypergraph, HyCLID designs a novel rethinking-based hypergraph neural network to learn the representation of interactions.
- Besides, we propose an adaptive mini-batch clustering method in the face of the large scale of the interaction data, which performs deep clustering in a mini-batch fashion based on the learned interaction representations.
- Experiments show that HyCLID significantly outperforms state-of-the-art methods across public and industrial datasets for clustering interactions. Furthermore, experiments on recommendation demonstrate the practical value of the cluster patterns discovered by our method in industrial applications.

2 Related Work

In this section, we first introduce the related clustering methods, and then discuss the hypergraph approaches for interaction data.

Clustering, as one of the fundamental tasks of data mining, is widely used for interaction data analysis. Traditional clustering studies are feature-based. They can target a certain type of interacting objects, and encode them as vectors based on their features [7] followed by traditional clustering methods such as K-means [6]. Besides, graph-based clustering is also a representative kind of clustering method, which treats objects as nodes in a graph and leverages structural information of objects for clustering the objects, e.g., spectral clustering [16]. Recently, attributed graph clustering has attracted a mass of attention, which further considers the attributes of nodes [28]. A noticeable trend is the wide adoption of deep learning in clustering [15]. For example, Graph Neural Network (GNN) [12, 11] based clustering methods can effectively learn representations based on the graph structure and node attributes for clustering [1, 21, 2, 18, 29]. However, modeling the interaction data using the graph structure, which can only model the pairwise relations, will inevitably lose some information since an interaction usually involves more than two objects [27].

Hypergraphs, as generalizations of graphs, can model complex and extensive information and be more suitable for interaction data [33, 25, 20, 8]. How to develop hypergraph structure based solutions for interaction data attracts increasing attention. [34] puts forward the concept of learning with hypergraphs, which can be used in clustering, classification, and embedding and achieves performance beyond the ordinary graphs. Hypergraph neural networks (HGNN) proposed in [3], which are similar to the GCNs in graphs [12], extend the convolution operation to the process of hypergraph learning. [35] proposes an unsupervised method to conduct both subspace

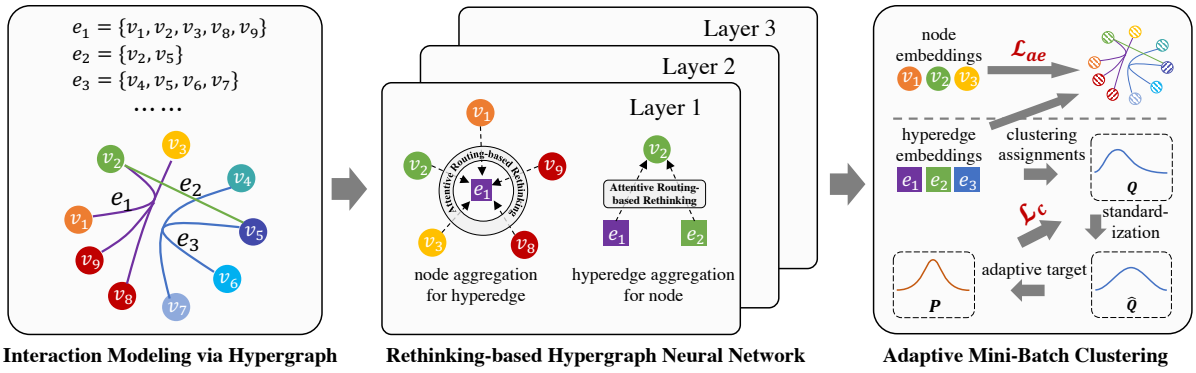


Figure 2: Illustration of HyCLID (object attributes and interaction conditions are omitted for easy to understand).

learning and feature selection, but its complex optimization details do not make it perform better than the above graph methods for clustering tasks. A recent study thereby proposes an end-to-end clustering method based on hypergraph neural network [8]. However, these methods only target the individual node/object, which breaks the intrinsic collaborative associations among the involved objects and conditions. The related works for clustering the interactions still are little exploration.

3 Methodology

First of all, we present an overview of the proposed HyCLID. The basic idea is that, as illustrated in Figure 2, with modeling interactions via attributed hypergraph, we propose a novel rethinking-based hypergraph neural network to learn the representations of interactions for clustering. Specifically, to model the interactions that involve multiple objects and conditions, we construct an attributed hypergraph, where each interaction is modeled as a hyperedge connecting several nodes that represent its involved objects. Besides, the attributes of objects and conditions are introduced as node features and hyperedge features, respectively. Then, a novel rethinking-based hypergraph neural network is proposed to aggregate node and hyperedge features to obtain interaction and object representations. During feature aggregation, we design a novel attentive routing-based rethinking mechanism, which can capture high-order features and then rethink to check for omissions and errors. Finally, in the face of the large scale of the interaction data, we develop an adaptive mini-batched clustering method to cluster in a mini-batch fashion based on the learned interaction representations. Moreover, it employs an adaptive batch standardization to remove the impact of different data distributions of different batches.

3.1 Interaction Modeling via Hypergraph

As mentioned above, an interaction usually involves multiple attributed objects and the corresponding interaction conditions such as temporal-spatial contexts, etc. We formalize the *interactions* as follows.

Definition 1: Interaction. An interaction is an action collaboratively done by several objects under a certain condition. Formally, given the set of objects $\mathcal{V}$, object attributes $A^{\mathcal{V}}$, and interaction conditions $A^{\mathcal{E}}$, an interaction is defined as $e =< \mathcal{V}_e, A^{\mathcal{V}}_e, A^{\mathcal{E}}_e >$, including all involved objects $\mathcal{V}_e = \{v_1, \cdots, v_{n_e} | v_i \in \mathcal{V}\}$ with their attributes $A^{\mathcal{V}}_e = \{a_1, \cdots, a_{n_e} | a_i \subset A^{\mathcal{V}}\}$, and interaction conditions $A^{\mathcal{E}}_e \subset A^{\mathcal{E}}$.

Take Interaction 1 in Figure 1 as an example, "a fifteen-year-old student ordered a middle-size latte at the Internet celebrity store Starbucks in the afternoon". In this example, the interaction *order* involves the objects including a user (with occupation attribute *student* and age attribute *fifteen*), a merchant (with attribute *Starbucks*

and *Internet celebrity*) and an item *coffee* (with attribute *latte* and *middle-size*). Besides, this interaction has a condition *afternoon*. In order to avoid the association breaking in traditional graph modeling, which splits the collaborative association among multiple objects into several pairwise sub-relations, we model the interactions as an attributed hypergraph.

In detail, each of the interactions is modeled as a hyperedge connecting several nodes that represent its involved objects. Moreover, object attributes are the features of corresponding nodes, and condition attributes are the features of the hyperedges. For example, as illustrated in left of Figure 2, for an interaction e_1 involving 5 objects v_1, v_2, v_3, v_8, v_9, we build a hyperedge to connect them. Besides, we attach their object attributes $a_i (i = 1, 2, 3, 8, 9)$ to the node features. Then the condition attributes $\breve{A}^{\mathcal{E}}_{e_1}$, such as temporal-spatial contexts, are attached to the hyperedge features, since they should be seen as the attributes of interactions rather than any object. Therefore, such a hyperedge and its connecting nodes together with their features can represent an instance of interactions. The constructed attributed hypergraph is formalized as follows.

Definition 2: Attributed Hypergraph for Interactions. The hypergraph for interactions is an attributed hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X}_V, \boldsymbol{X}_E)$, where the node set $\mathcal{V}$ represents the set of all involved objects, the hyperedge set $\mathcal{E}$ represents all the interactions, $\boldsymbol{X}_V$ and $\boldsymbol{X}_E$ represent the feature matrices of nodes and hyperedges, respectively.

The i-th row of $\boldsymbol{X}_V \in \mathbb{R}^{|\mathcal{V}| \times F_V}$ represents the feature indicators to the attributes $a_i \subset A^{\mathcal{V}}$ of object $v_i \in \mathcal{V}$. Similarly, the j-th row of $\boldsymbol{X}_E \in \mathbb{R}^{|\mathcal{E}| \times F_E}$ represents the feature indicators to the condition attributes $A^{\mathcal{E}}_{e_j} \subset A^{\mathcal{E}}$ of interaction $e_j \in \mathcal{E}$. F_V and F_E are the max value across the attribute number of each object and condition, respectively.

Based on the above modeling, the clustering task of interactions in this paper can be formalized as the clustering on hyperedges $\mathcal{E}$ in the constructed attributed hypergraph, which is different from other clustering methods that mainly concern about clustering of objects $\mathcal{V}$ in interactions [8].

3.2 Rethinking-based Hypergraph Neural Network

To learn the representation of interactions, we propose a novel rethinking-based hypergraph neural network, where an attentive routing-based rethinking mechanism is designed to capture high-order features and avoid omissions. It can automatically identify the correlations of features and form some meaningful combinations of high-order features. Then, it rethinks the identification process several times to prevent omissions and correct errors. Consequently, the network can learn effective representation of interactions.

3.2.1 Input Layer.

In order to facilitate capturing high-order features, following [19], we initial the representations of each feature from hyperedges or nodes as vectors, thus the representations of hyperedges and nodes are formed as feature matrices. Formally, the initial d-dimensional representations of hyperedge $e_i \in \mathcal{E}$ and node $n_j \in \mathcal{N}$ are $\boldsymbol{e}_i^{(0)} \in \mathbb{R}^{F_E \times d}$ and $\boldsymbol{v}_j^{(0)} \in \mathbb{R}^{F_V \times d}$, where each row of the feature matrix denotes a specific attribute of the node/hyperedge.

3.2.2 Layer-wise Aggregation.

For the hypergraph $\mathcal{G}$, the incidence matrix is defined as $\boldsymbol{H} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$ with entries $h(v, e) = 1$ if $v \in e$ and 0 otherwise. Denote the diagonal matrices of the edge degrees, the node degrees and the pre-defined weights of hyperedges (default is 1) as $\boldsymbol{D}_e$, $\boldsymbol{D}_v$ and $\boldsymbol{W}$, respectively. The spectral hypergraph convolution for node embedding [3, 8] is formulated as

$$\boldsymbol{Y} = \boldsymbol{D}_v^{-1/2} \boldsymbol{H} \boldsymbol{W} \boldsymbol{D}_e^{-1} \boldsymbol{H}^\top \boldsymbol{D}_v^{-1/2} \boldsymbol{X}_V \boldsymbol{\Theta}, \tag{32}$$

where $\mathbf{\Theta}$ is the trainable filter parameters and $\boldsymbol{X}_V$ is the input node embeddings. It can be seen that in this hypergraph convolution formula, neither can the features of the hyperedge be exploited, nor can the representation of the hyperedges (interactions) be obtained.

To solve these problems, we analyze this formula intuitively and find that this form can be also understood as the following information flow: The node embeddings are first aggregated into hyperedge embeddings by multiplying matrix $\boldsymbol{H}^\top$. Then the hyperedge information is passed back to the nodes, achieved by multiplying $\boldsymbol{H}$. We can formalize these two information aggregation processes as a two-stage aggregation rule from layer l to layer $l+1$ as follows.

$$\boldsymbol{e}_i^{(l+1)} = \text{aggr}(\boldsymbol{e}_i^{(l)}, \{\boldsymbol{v}_j^{(l)} \,|\, v_j \in e_i\}), \tag{33}$$

$$\boldsymbol{v}_j^{(l+1)} = \text{aggr}(\boldsymbol{v}_j^{(l)}, \{\boldsymbol{e}_i^{(l+1)} \,|\, v_j \in e_i\}). \tag{34}$$

For the aggregation function $\text{aggr}(\cdot)$, we first concatenate all its inputs into an whole feature matrix $X = \text{concat}(\boldsymbol{e}_i^{(l)}, \{\boldsymbol{v}_j^{(l)} | v_j \in e_i\})$ or $X = \text{concat}(\boldsymbol{v}_j^{(l)}, \{\boldsymbol{e}_i^{(l+1)} |$ $v_j \in e_i\})$, then we propose the following module to capture the high-order features based on X before obtaining the node/hyperedge embeddings. Please note that the size of X is not fixed. The number of its rows depends on the number and size of the concatenated feature matrices of hyperedges and nodes. For example, the size of the whole feature matrix for Eq. 33 is $(F_E + |e_i| \cdot F_V) \times d$. Denoting the calculation of this module as function $\text{ARR}(\cdot)$ for short, the aggregation function can be formalized as $\text{aggr}(\cdot) = \text{ARR}(\text{concat}(\cdot))$.

3.2.3 Attentive Routing-based Rethinking Mechanism

As introduced before, the interactions often arise from the intrinsic correlations among several related attributes. Motivated by previous studies [19, 24], the correlations among p attributes are formalized as a p-order features. As shown in Figure 3, we adopt the framework of multiple self-attentive layers to capture high-order features. Furthermore, in order to prevent omissions of potential high-order features, we need to review the capturing process to check for omissions and errors. Consequently, we introduce a routing mechanism and modify the self-attentive layers to rethink and readjust the captured high-order features. The details are introduced as follows.

Following [19, 24], we adopt the framework of self-attention to learn high-order features due to its superiority for saving computational and storage. However, it needs further improvement to support rethinking. Specifically, we introduce a rethinking vector $\hat{z}$, the attention score between feature x_m and x_k, i.e., the m-th and k-th rows of the input feature matrix X, is calculated as,

$$\varphi(\hat{z}; x_m, x_k) = \text{sum}\left(\hat{z} \odot W_{\text{Q}} x_m \odot W_{\text{K}} x_k\right), \tag{35}$$

where $\odot$ represents element-wise product, i.e., Hadamard product. W_{Q} and W_{K} denote the transformations for Query vector and Key vector in attention. Then the group of second-order features related to feature x_m can be combined together through the framework of self-attention as follows.

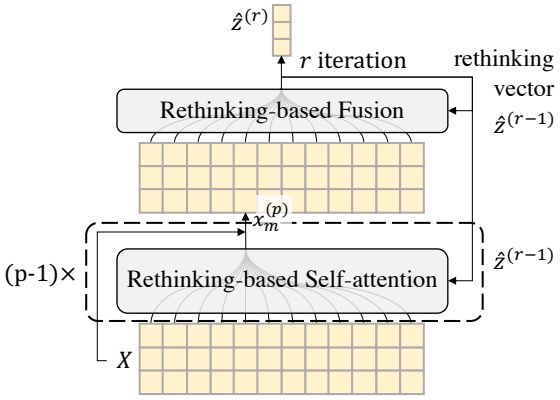


Figure 3: Illustration of the attentive routing-based rethinking mechanism.

$$\alpha_{m,k} = \exp\left(\varphi(\hat{z}; x_m, x_k)\right) / \sum\nolimits_{j=1} \exp\left(\varphi(\hat{z}; x_m, x_j)\right), \tag{36}$$

$$x_m^{(2)} = x_m + \sum\nolimits_{k=1} \alpha_{m,k} (W_{\text{V}} e_k), \tag{37}$$

where W_{V} denotes the transformation for Value vector in attention. Next, a higher-order feature $x_m^{(p)}$ related to x_m can be captured by stacking $p-1$ layers of such self-attention with residue connections. Eventually, the final combination $\hat{z}^{(1)}$ of feature of order p can be obtained by attentive fusion of $x_m^{(p)}, m = 1, 2, \cdots,$

$$\hat{z}^{(1)} = \sum\nolimits_m \text{SoftMax}\left((\hat{z} \odot \boldsymbol{x}_m^{(p)})^\top \cdot \boldsymbol{a}\right) \cdot \boldsymbol{x}_m^{(p)}\,, \tag{38}$$

where $\boldsymbol{a}$ is the vector of attention parameters.

Furthermore, to prevent omissions of potential high-order features, we introduce a routing mechanism to rethink and readjust the above capturing process. Specifically, we update the rethinking vector by the new obtained $\hat{z}^{(1)}$, and repeat the above procedure several times until routing-by-agreement, which is called attentive routing-based rethinking mechanism. Therefore, each repetition of the above process, i.e., each iteration of the routing mechanism, refines the captured high-level features. Initially, let $\hat{z} = \text{mean}_m(W_{\mathrm{V}} x_m)$.

The output high-order feature after repeating r times is denoted as $\hat{z}^{(r)}$, which is the output vector of this module. Formally, given an input feature matrix X, we have $\text{ARR}(X) = \hat{z}^{(r)}$. Then, the Eq. (33) can be rewritten by $\boldsymbol{e}_i^{(l+1)} = \text{ARR}(X)$, where X is obtained by concatenating the embedding matrices of a target hyperedge and its connecting nodes.

3.2.4 Interaction & Object Embedding.

Since each layer of propagation represents a specific order[2] of relations, we sum the embeddings from each layer as the final representation both for hyperedges (interactions) and nodes (objects). Formally, we have $\boldsymbol{e}_i = \sum_l^L \boldsymbol{e}_i^{(l)}$ and $\boldsymbol{v}_j = \sum_l^L \boldsymbol{v}_j^{(l)}$, where L denotes the number of aggregation layers.

3.3 Adaptive Mini-batch Clustering

For clustering in a mini-batch fashion, the different data distributions of batches may lead to clustering errors [26], thus making clustering difficult to discover the implicit patterns of interaction data. Therefore, enlightened by existing deep clustering methods [23], we propose an adaptive mini-batch clustering method, which first removes the impact of different data distributions via an adaptive batch standardization and then performs deep clustering on interactions.

Specifically, after the above modules, we have obtained the representations of interactions. Enlightened by Clustering Assignment Hardening [23], we take Student's t-distribution as a kernel to measure the similarity between points and centroids. Then the soft assignment matrix Q is formulated as follows:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\nu)^{-\frac{\nu+1}{2}}}{\sum_{j'}(1 + \|z_i - \mu_{j'}\|^2/\nu)^{-\frac{\nu+1}{2}}}, \tag{39}$$

where z_i is the embedding of the i^{th} interaction, μ_j is the j^{th} cluster centroid, and ν is the degrees of freedom of the Student's t-distribution, e.g. $\nu = 1$.

Due to the randomness of batch generating process, the assignment distribution Q can be very unstable. It severely limits the robustness of clustering, which in turn brings difficulties to discovering patterns in interaction data. To tackle this issue, we first square the Q distribution to speed up training and then batch standardize it to obtain $\hat{Q}$, which removes the influence of batch quality on the clustering performance. Formally,

$$\widehat{q}_{iu} = \left(q_{iu}^2 - \mu_u\right) / \left(\sqrt{\sigma_u^2 + \epsilon}\right), \tag{40}$$

$$\text{where} \quad \mu_u = \frac{1}{m}\sum\nolimits_i q_{iu}^2 \quad \text{and} \quad \sigma_u^2 = \frac{1}{m}\sum\nolimits_i \left(q_{iu}^2 - \mu_u\right)^2. \tag{41}$$

[2]Please note that the order of relations is the path length between neighbors and the target node, while the order of features is the number of features combined into high-order feature.

Table 16: Statistics of datasets.

Dataset	# Nodes						# Hyperedges	# Attributes	# Categories	Graph Density
ACM	Paper	4,025	Author	7,167	Field	60	4,025	1,902	3	$\sim 10^{-4}$
IMDB	Movie	4,661	Director	5,841	Actor	2,270	4,661	1,256	3	$\sim 10^{-4}$
MT-S	User	19,623	POI	6,489	Item	17,564	20,000	901	5	$\sim 10^{-5}$
MT-L	User	991,914	POI	17,120	Item	389,292	2,000,000	901	5	$\sim 10^{-6}$

Note that this standardization operation makes $\hat{Q}$ appear negative numbers. It means that we can just erase the low-probability cluster assignment via ReLU function, which contributes to to reduce the difficulty of clustering training and improve performance. Furthermore, we introduce two batch adaptive trainable parameters γ, β to enhance its adaptive ability as follows.

$$p_{iu} = \frac{\text{ReLU}(\gamma_u \widehat{q}_{iu} + \beta_u)}{\sum_s \text{ReLU}(\gamma_s \widehat{q}_{is} + \beta_s)}, \qquad s.t. \sum\nolimits_u \gamma_u = 1, \tag{42}$$

where γ represents the relative stability between different clusters (small value means that the samples of one cluster are prone to be modified to another cluster during training), and β defines the aforementioned low-probability cluster assignment. Consequently, the adaptive target distribution $P = [p_{iu}]$ forces assignments to have stricter probabilities (closer to 0 or 1). Meanwhile, we not only preserve the stability of the Q distribution by $\hat{Q}$, but also increase its adaptive capacity. Finally, KL-divergence is applied to let the raw assignments Q approach the target distribution P. which can be minimized for the aforementioned Q and P via neural network training. It emphasizes data points assigned with batch-adaptive high confidence.

3.4 Model Training

We apply an auto-encoder structure for self-supervised training of our HyCLID. Specifically, we reconstruct the incidence matrix H of the constructed hypergraph for interactions with a contrastive loss:

$$\mathcal{L}_{ae} = \frac{1}{2} \sum_{i,j} y_{ij} d_{ij}^2 + (1 - y_{ij}) \max(0, m - d_{ij})^2, \quad \text{where} \quad d_{ij} = \|\boldsymbol{e}_i - \boldsymbol{v}_j\|. \tag{43}$$

y_{ij} denotes the existence of a relationship between hyperedge i and node j, and m is the margin hyper-parameter, e.g., $m = 1$.

Finally, we jointly optimize the auto-encoder structure and deep clustering so that the total objective function is defined as $\mathcal{L} = \mathcal{L}_c + \gamma \mathcal{L}_{ae}$, where $\gamma \geqslant 0$ is a coefficient that controls the balance in between.

4 Experiments

4.1 Experimental Setup

4.1.1 Datasets

Our proposed HyCLID is evaluated on the following four datasets, and the statistics of these datasets are shown in Table 16.

- **ACM.** The ACM dataset[3] contains three types of nodes: paper, author and field. We treat a publishing as an interaction, and hence build a hypergraph where each hyperedge connects a paper and all its corresponding authors and fields. Considering there are no labels on the hyperedges, we treat the category of the published paper as the clustering ground truth since each paper has a one-to-one correspondence to a hyperedge.

[3] https://data.dgl.ai/dataset/ACM.mat

- **IMDB.** The IMDB dataset [31] contains three types of nodes: movie, actors and directors. Similarly, we treat each movie as an interaction and construct a hypergraph where each hyperedge connects a movie, its actors and directors. The category of the movie is regarded as the clustering ground truth.
- **MT-L** and **MT-S.** We also build two real-world datasets from the food delivery industry, i.e., Meituan Waimai platform. One contains millions of orders of user purchases of foods in Beijing District within 30 days, denoting as MT-L. Each purchase order is treated as an interaction and is tagged with one of 5 purchase scenes. We construct an attributed hypergraph where each hyperedge represents an order, connecting a user node, a restaurant node and several item nodes. The features of nodes are their attributes (e.g., user profiles for user node), while the features of hyperedges (orders) are the interaction conditions, i.e., temporal-spatial contexts, since these attributes do not belong to a single node. Most baseline methods cannot support data of such scale, so we extracted a smaller dataset from it, denoted as MT-S.

4.1.2 Evaluation Metrics

Following [8], we adopt three popular metrics to assess the quality of the clustering results: clustering accuracy (ACC), normalized mutual information (NMI) and adjusted Rand index (ARI). For all the experiments, we repeat them 10 times and report the averaged results and standard deviations.

4.1.3 Baselines

We compare our proposed method HyCLID with the following three groups of methods. Traditional Methods: *K-means* [6], *node2vec* [4] and *HERec* [17]. Attributed Graph-based Methods: *SDCN* [1], *HAN* [22], *HGT* [9] and *AdaGAE* [14]. Hypergraph-based Methods: *HGNN* [3] and *AHGAE* [8]. Note that for the methods considering graph structure, for ACM and IMDB datasets, we directly use the widely applied graph structure [9]. For the MT-S dataset, we transfer the hypergraph into a graph, i.e., we introduce a summary node to replace each of hyperedges and link the summary nodes with the corresponding nodes that are originally connected by hyperedges. For metapath-based methods, i.e., HERec and HAN, we select all the possible symmetric length-2 meta-paths.

4.1.4 Implementation Detail

We implement the proposed method based on Tensorflow[4]. For our method, we set the dimension of attribute embeddings as $d = 64$ for the public datasets ACM and IMDB for fair comparison and $d = 16$ for MT-S/L for saving memory. For simplicity, we set the number of aggregation layers $L = 1$, the layer number of rethinking-based self-attention $p = 1$ and the rethinking iterations $r = 2$. For all baselines, we set their hidden dimensions as 64 and set the number of layers as their suggested value (usually equal to 2). For model training, we simply set $\gamma = 1$, and apply Adam [10] to optimize with the learning rate as 0.005 for ACM and IMDB and 0.001 for MT-S/L. The batch size is set 2048. All the experiments are performed in NVIDIA Tesla P40 Cluster. To facilitate related research, we will release our implementation to the public once the paper gets accepted.

4.2 Analysis of Clustering Results

Table 17 shows the clustering results on the three small-scale datasets. We have the following observations:

As shown, for each metric, our proposed HyCLID achieves the best results significantly. In particular, compared with the best results of the baselines, our approach achieves a significant improvement of 6.3% on ACC, 17.9% on NMI, 15.9% on ARI averagely. The reason is that HyCLID successfully makes full use of all the information in interactions, including node attributes, hyperedge attributes and hyper-relations. In detail, attributed

[4]https://www.tensorflow.org/

Table 17: Clustering results on datasets ACM, IMDB and MT-S (mean±std in percent). The best and second best results are bold and underlined, respectively. Symbol "-" represents unavailable results due to out-of-memory.

Method	ACM			IMDB			MT-S		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
K-means	63.59 ± 0.27	36.89 ± 0.37	28.91 ± 0.42	37.01 ± 0.10	0.88 ± 0.06	1.43 ± 0.05	54.95 ± 8.08	56.36 ± 6.69	50.08 ± 7.83
node2vec	65.25 ± 0.33	38.49 ± 0.12	31.23 ± 0.88	47.87 ± 0.33	5.55 ± 0.53	5.89 ± 0.64	45.09 ± 6.70	23.19 ± 5.60	18.65 ± 7.78
HERec	51.41 ± 1.82	20.05 ± 1.37	20.82 ± 1.50	46.45 ± 0.62	5.32 ± 0.79	5.68 ± 0.69	39.76 ± 9.72	20.57 ± 2.04	18.32 ± 3.11
SDCN	67.79 ± 0.87	41.77 ± 0.80	37.69 ± 0.60	43.87 ± 0.35	3.37 ± 0.19	2.77 ± 0.21	73.56 ± 9.27	66.34 ± 8.02	62.80 ± 7.83
HAN	69.33 ± 0.38	44.17 ± 0.58	38.94 ± 0.36	42.43 ± 0.32	2.90 ± 0.37	2.89 ± 0.29	55.71 ± 5.96	54.09 ± 6.41	48.34 ± 8.25
HGT	75.31 ± 1.17	46.98 ± 2.28	43.06 ± 1.63	48.07 ± 0.02	5.62 ± 0.06	5.23 ± 0.05	55.22 ± 8.13	57.38 ± 7.62	51.11 ± 9.27
AdaGAE	78.32 ± 3.76	50.94 ± 3.12	54.03 ± 4.09	47.12 ± 3.17	5.07 ± 1.86	4.58 ± 1.73	-	-	-
HGNN	66.35 ± 2.29	31.17 ± 4.05	27.72 ± 5.32	44.11 ± 0.11	2.05 ± 0.01	1.31 ± 0.04	80.75 ± 4.32	76.70 ± 5.83	68.46 ± 3.91
AHGAE	75.76 ± 1.83	46.53 ± 2.98	41.28 ± 3.64	40.05 ± 1.54	1.73 ± 1.35	1.58 ± 1.07	84.66 ± 8.01	79.16 ± 8.98	71.53 ± 9.23
HyCLID	**83.63 ± 1.23**	**57.29 ± 0.76**	**58.45 ± 1.91**	**50.28 ± 0.39**	**7.43 ± 0.63**	**7.13 ± 1.02**	**91.05 ± 8.61**	**86.30 ± 6.24**	**84.70 ± 9.44**
Impr. (%)	6.78	12.46	8.18	4.60	32.20	21.05	7.55	9.02	18.41

graph-based methods can generally achieve better performance than traditional methods, especially heterogeneous GNN-based end-to-end methods (HGT), which confirms the important role of object attributes and structures among objects in the interaction data. Unexpectedly, hypergraph-based methods (HGNN, AHGAE) have not achieved competitive performance on these two datasets, while our method HyCLID successfully performs the best. We hold that this is because these hypergraph-based methods cannot capture high-order correlations (as our rethinking-based self-attention) nor be adaptive during clustering, which makes their hypergraph neural networks fail to fully utilize the information in the interaction data. Conversely, the hypergraph-based baselines achieve outstanding performance on the MT-S dataset. It suggests the necessity of attributed hypergraph modeling for real-world interaction data, which involves multiple attributed objects and a certain interaction condition. Besides, the fact that our HyCLID further performs better confirms again the effectiveness of our proposed attentive routing-based rethinking mechanism to capture high-order features and the adaptive mini-batch clustering for training.

Moreover, we also conduct a comparison on the MT-L dataset. Since most baselines cannot support running on such a large-scale dataset, we only employ mini-batch K-Means for comparison. As shown in Table 18, our proposed HyCLID consistently performs better than the baseline, which demonstrates the effectiveness of our proposed HyCLID for large-scale data.

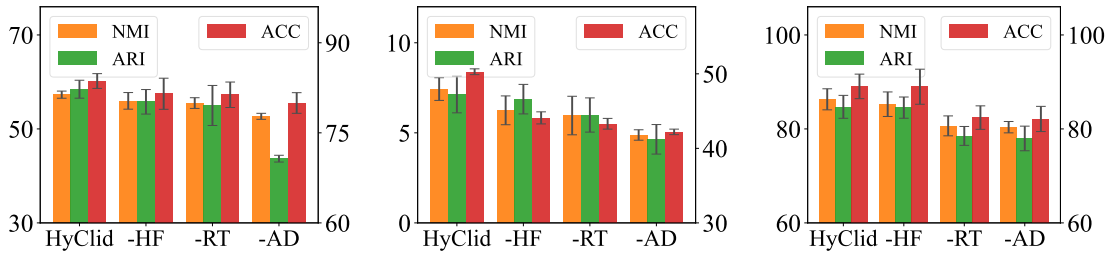
Table 18: Clustering results on dataset MT-L.

Method	ACC	NMI	ARI
Mini-batch K-means	45.82 ± 3.75	24.48 ± 2.32	18.30 ± 3.88
HyCLID	**48.60 ± 3.98**	**28.68 ± 4.92**	**24.50 ± 5.95**

4.3 Ablation Study

In this subsection, for simplicity, we compare our HyCLID with 3 variants on the datasets ACM, IMDB and MT-S to validate the design of each module. Specifically, *-HF* is a variant that removes the part to capture high-order features (i.e., set $p = 0$) based on the complete model. The variant *-RT* represents that the module attentive routing mechanism for rethinking is removed, i.e., the vanilla self-attention is adopted for replacement. *-AD* denotes that the adaptive mini-batch clustering is further removed and replaced by traditional deep clustering, i.e., clustering assignment hardening [23].

As reported in Figure 4, the clustering performance always decreases with the removal of each module. This phenomenon demonstrates the effectiveness of each of the proposed modules. In detail, compare the best performance achieved by our whole model, *-HF* performs worse, which verifies the existence of the correlations



(a) ACM (b) IMDB (c) MT-S

Figure 4: Clustering results of different variants.

among multiple attributes from the involved objects and conditions and confirms the effectiveness of our rethinking based self-attention to capture these correlations. Next, the fact that *-RT* performs more poorly shows the necessity of our rethinking module for avoiding the omissions of some potential high-orders. Furthermore, when the adaptive mini-batch clustering is further substituted by the traditional clustering assignment hardening [23], *-AD* performs the worst. On the one hand, it shows that problems such as data imbalance make the clustering significantly worse, and on the other hand, it demonstrates that our proposed adaptive module can alleviate this problem very well.

4.4 Case Study for Rethinking Mechanism

In this subsection, we dissect how our attentive routing-based rethinking mechanism works. As shown in Figure 5, the heat maps visualize the attention weights of rethinking-based fusion (top) and rethinking-based self-attention (bottom) under two iterations of rethinking (three pairs of heat maps from left to right represent $r = 0, 1, 2$, respectively).

Figure 5: Heat map visualization for attentive routing-based rethinking mechanism. The rows/columns represent four selected attributes from an interaction case belonging to category "afternoon tea" of dataset MT-S, i.e., *lower consumption level*, *student*, *coffee*, *afternoon*, respectively.

A phenomenon can be observed that, the self-attention assigns relatively high weights to the second-order feature between *coffee* and *afternoon*, while the fusion attention also assigns high weights to *coffee* and *afternoon*. Then, the weights are further growing with the iterations of rethinking. On the contrary, the high weight, assigned to the second-order feature between *lower consumption level* and *afternoon*, is gradually reduced. It validates the effectiveness of our attentive routing-based rethinking mechanism. In detail, it not only encourages the important features or higher-order features through iterative rethinking, but also corrects the overestimated ones.

4.5 Offline Experiment on Recommendation

In this subsection, we conduct an offline experiment on recommendation to show the applied usage and practical value of our method for food delivery industrial applications, i.e., Meituan Waimai platform. Specifically, we divide the MT-L dataset into training/test subsets, i.e., the orders of the first 27 days are for training while those of the last 3 days are for testing. Then we train our model on the training subset of MT-L and then infer

the clustering assignments on the test subset. Finally, these learned and inferred clustering assignments are introduced into context-aware recommendation models, e.g., DeepFM [5] and AutoInt [19], as contexts for CTR prediction, where the dataset partition is the same as above. For comparisons, "-vanilla" represents that no clustering assignments are introduced, and "+HyCLID" represents that our HyCLID is adopted for assignments.

We report the results on several metrics (including HR, Recall, Precision and NDCG) in Table 19. As we can see, for both DeepFM and AutoInt, "+HyCLID" outperforms "-vanilla", indicating that the clustering results of interactions are conducive for downstream applications and demonstrates the necessity of our attributed hypergraph modeling of interactions and the effectiveness of our proposed hypergraph clustering network for clustering interactions.

Table 19: Top-3 recommendation performance.

Method	HR	Recall	Precision	NDCG
DeepFM-vanilla	0.5268	0.4950	0.1849	0.4135
DeepFM+HyCLID	**0.5386**	**0.5056**	**0.1868**	**0.4186**
AutoInt-vanilla	0.5120	0.4811	0.1786	0.4021
AutoInt+HyCLID	**0.5465**	**0.5142**	**0.1906**	**0.4236**

5 Conclusion

In this paper, we propose a novel clustering task on interaction data to discover more comprehensive cluster patterns, where the interactions are regarded as basic units to be clustered instead of objects. To this end, we propose a novel Hypergraph CLustering network for Interaction Data, namely HyCLID. Specifically, we propose to model the interactions via attributed hypergraph, and then propose a novel rethinking-based hypergraph neural network to learn the representation of interactions. It designs a novel attentive routing-based rethinking mechanism to capture the correlations among multiple attributes of objects and conditions involved in interactions. Besides, we develop an adaptive mini-batch clustering method to deal with the large scale of interaction data, which can further make adaptive adjustments for different data distributions of batches. Extensive experiments verify the effectiveness of our HyCLID on both public datasets and real industrial datasets. Moreover, offline experiments on recommendation show the practical value of our discovered cluster patterns for industrial applications.

References

[1] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural Deep Clustering Network. In WWW, 2020.

[2] Shaohua Fan, Xiao Wang, Chuan Shi, Emiao Lu, Ken Lin, and Bai Wang. One2Multi Graph Autoencoder for Multi-view Graph Clustering. In WWW, pages 3070–3076. ACM, April 2020.

[3] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph Neural Networks. In AAAI, pages 3558–3565, July 2019.

[4] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. In SIGKDD, pages 855–864. Association for Computing Machinery, 2016.

[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. In IJCAI, pages 1725–1731, 2017.

[6] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. Applied Statistics, 28(1):100, 1979.

[7] G. E. Hinton. Reducing the Dimensionality of Data with Neural Networks. Science, 313(5786):504–507, July 2006.

[8] Youpeng Hu, Xunkai Li, Yujie Wang, Yixuan Wu, Yining Zhao, Chenggang Yan, Jian Yin, and Yue Gao. Adaptive Hypergraph Auto-Encoder for Relational Data Clustering. TKDE, 2021.

[9] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous Graph Transformer. In WWW, pages 2704–2710. ACM, 2020.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

[11] Thomas N Kipf and Max Welling. Variational Graph Auto-Encoders. In NIPS Workshop on Bayesian Deep Learning, 2016.

[12] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR. OpenReview.net, 2017.

[13] Feifei Kou, Junping Du, Yijiang He, and Lingfei Ye. Social network search based on semantic analysis and learning. CAAI Transactions on Intelligence Technology, 1(4):293–302, October 2016.

[14] Xuelong Li, Hongyuan Zhang, and Rui Zhang. Adaptive Graph Auto-Encoder for General Data Clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

[15] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. IEEE Access, 6:39501–39514, 2018.

[16] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. NIPS, 14, 2001.

[17] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. Heterogeneous information network embedding for recommendation. IEEE TKDE, 31(2):357–370, 2018.

[18] Hanyu Song, Peizhao Li, and Hongfu Liu. Deep Clustering based Fair Outlier Detection. In SIGKDD 2021, June 2021.

[19] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In CIKM, pages 1161–1170. ACM, 2019.

[20] Loc Hoang Tran and Linh Hoang Tran. Directed hypergraph neural network. arXiv, August 2020.

[21] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In IJCAI, August 2019.

[22] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. Heterogeneous Graph Attention Network. In WWW, 2019.

[23] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised Deep Embedding for Clustering Analysis. In ICML, volume 48, pages 478–487. PMLR, June 2016.

[24] Yichen Xu, Yanqiao Zhu, Feng Yu, Qiang Liu, and Shu Wu. Disentangled Self-Attentive Neural Networks for Click-Through Rate Prediction. In CIKM, pages 3553–3557. ACM, 2021.

[25] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In NeurIPS, pages 1511–1522. 2019.

[26] Yilin Yan, Min Chen, Mei-Ling Shyu, and Shu-Ching Chen. Deep Learning for Imbalanced Multimedia Data Classification. In IEEE ISM, pages 483–488, 2015.

[27] Chaoqi Yang, Ruijie Wang, Shuochao Yao, and Tarek Abdelzaher. Hypergraph learning with line expansion. arXiv preprint arXiv:2005.04843, 2020.

[28] Renchi Yang, Jieming Shi, Yin Yang, Keke Huang, Shiqi Zhang, and Xiaokui Xiao. Effective and Scalable Clustering on Massive Attributed Graphs. In WWW. ACM, April 2021.

[29] Shuiqiao Yang, Sunny Verma, Borui Cai, Jiaojiao Jiang, Kun Yu, Fang Chen, and Shui Yu. Variational Co-embedding Learning for Attributed Network Clustering. arXiv, 2021.

[30] Tianchi Yang, Cheng Yang, Luhao Zhang, Chuan Shi, Maodi Hu, Huaijun Liu, Tao Li, and Dong Wang. Co-clustering Interactions via Attentive Hypergraph Neural Network. In SIGIR, pages 859–869. ACM, July 2022.

[31] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph Transformer Networks. In NIPS, volume 32. Curran Associates, Inc., 2019.

[32] Rui Zhang, Hongyuan Zhang, and Xuelong Li. Maximum Joint Probability With Multiple Representations for Clustering. IEEE TNNLS, pages 1–11, 2021.

[33] Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-SAGNN: a self-attention based graph neural network for hypergraphs. In ICLR, 2020.

[34] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with Hypergraphs: Clustering, Classification, and Embedding. In NeurIPS, pages 1601–1608, 2006.

[35] Xiaofeng Zhu, Yonghua Zhu, Shichao Zhang, Rongyao Hu, and Wei He. Adaptive Hypergraph Learning for Unsupervised Feature Selection. In IJCAI, 2017.

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

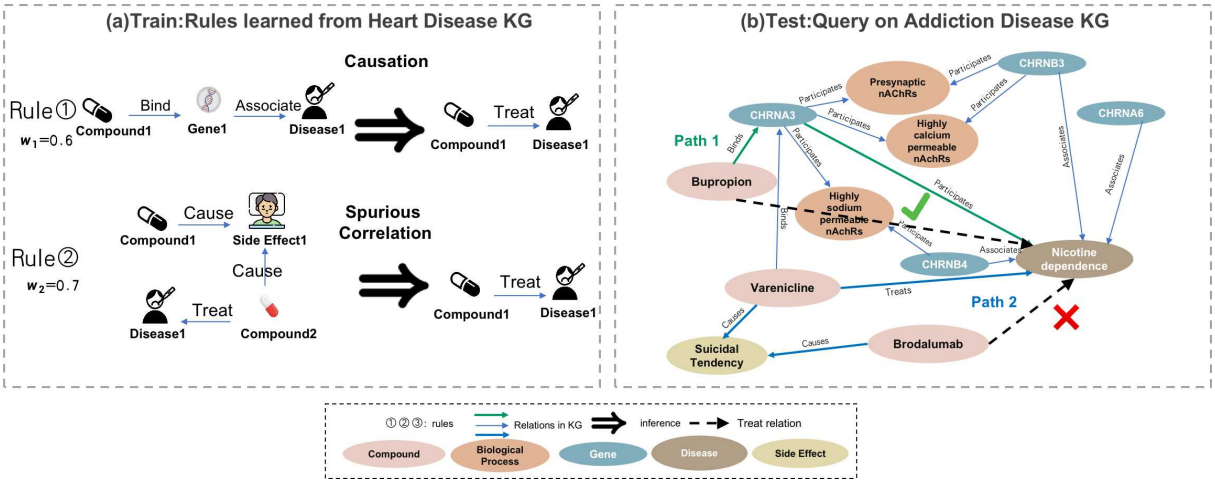


Figure 1: Motivation illustration. Consider a scenario to learn rules for inferring the `Treat` relation between compounds and diseases on a heart disease KG (top), and thereby discover novel drugs for treating nicotine addiction on an addiction disease KG (bottom). On the heart disease KG, drugs that 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.

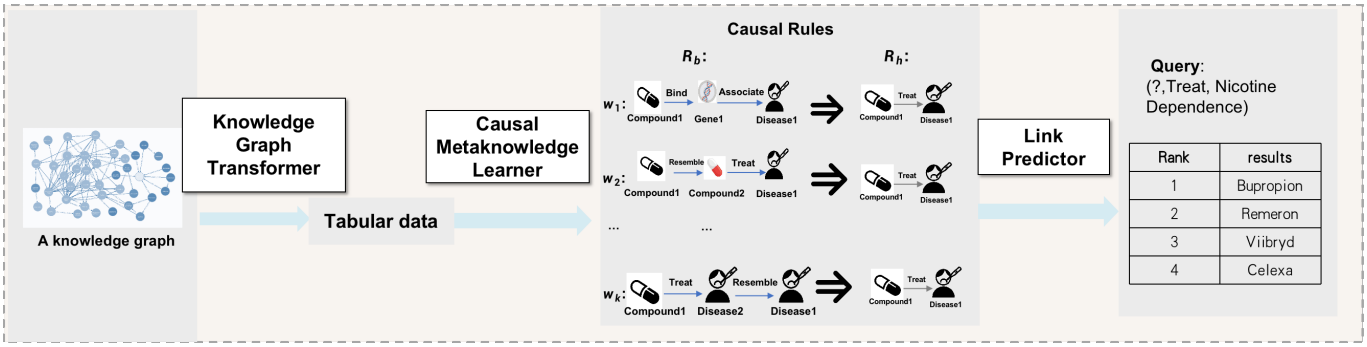


Figure 2: The framework of CMLP. Particularly, CFLP first transforms the relational data into propositional data for better statistical analysis. Then it mines interpretable causal rules, which can be interpreted as a kind of metaknowlege[6]. Finally, a plausibiliy score derived from the causality test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

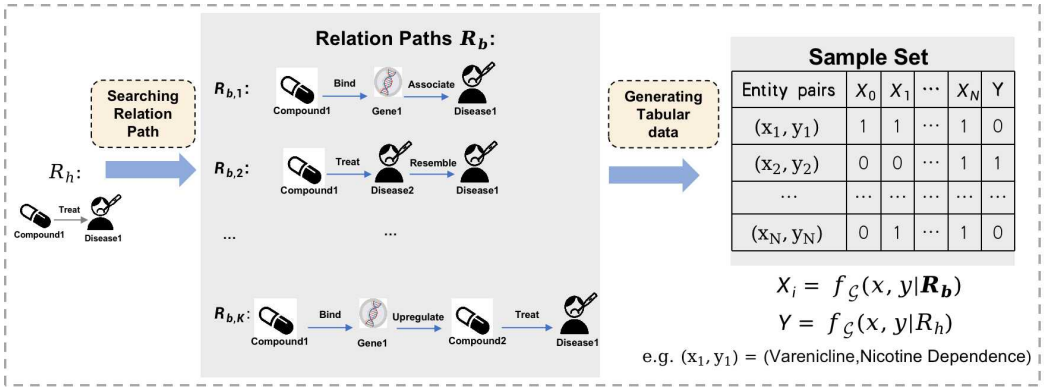


Figure 3: The process of knowledge graph transformer

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 $A \rightarrow B \rightarrow C$ or a fork $A \leftarrow B \rightarrow C$ such that the middle node B is in Z (i.e., B is conditioned on), or:

2. p contains a collider $A \rightarrow B \leftarrow C$ 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 ← 0;
while d <= |X^Ca| − 1 do
    for each X_k ∈ X^Ca do
        for each subset Z ∈ X^Ca\{X_k} and |Z| = d do
            Test CI(X_k, Y|Z);
            if CI(X_k, Y|Z) then
                Test CI(Z, Y|X_k) (Reverse CI test.) ;
                if not CI(Z, Y|X_k) then
                    Remove X_k from X^Ca;
                    Break;
                end
            end
        end
    end
    d ← d + 1;
end
X^C = 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. 1) 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

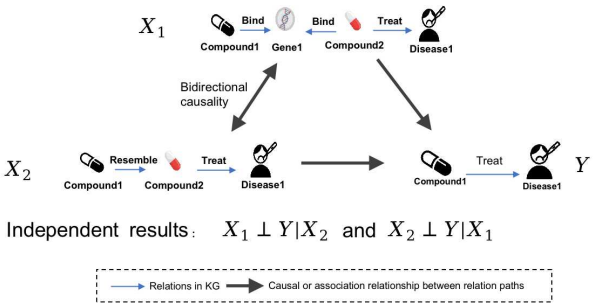


Figure 4: An example of bidirectional causal relationship, which may lead wrong results.

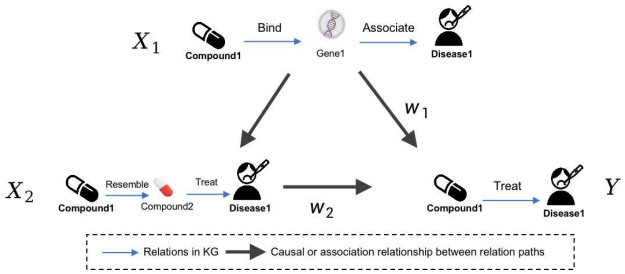


Figure 5: An example of non-independent rule-induced variables, which are both the causes of queried relation.

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. 1 (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. 1, 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 $X_1 \to X_2 \to Y$. 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:

$$\begin{aligned} w_j = \min(\{dependence(X_j, Y|\mathcal{Z})\}) \\ \text{for any subset } \mathcal{Z} \in \mathcal{X}^{Ca} \backslash \{X_j\}, \end{aligned} \tag{44}$$

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. 44, 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 $(?, R_h, e_t)$:

$$S_{R_q}^{sum} = \sum_{i=1}^{K} \tilde{w}_i Q_i, . \tag{45}$$

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. 45 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:

$$S_{R_q}^{avg} = \sum_{i}^{K} \tilde{w}_i \big(Q_i \bar{Y}_{X_i=1} + (1 - Q_i)\bar{Y}_{X_i=0}\big), \tag{46}$$

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. 46 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 20: 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 20.

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 21. In the out-of-distribution paragraph, we will introduce the parameters of root nodes in training and testing.

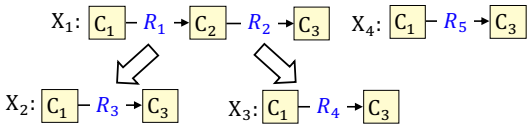


Figure 6: The causal graph of relation paths, based on which the simulated KGs are generated .

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 21: 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 20. 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].

$$MRR = \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{q} \tag{47}$$

$$Hit@K = \frac{|\{q \in Q : q \leq K\}|}{|Q|} \tag{48}$$

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].

$$Precision = \frac{\#TFR}{\#FR}; Recall = \frac{\#TFR}{\#ATR} \tag{49}$$

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 23 and Table 24.

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 22 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. 23 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 22: 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. 24 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 25 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 26 (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 23: 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 24: 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 25: 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 26: 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) ~~$R_3(C_1, C_3) \leftarrow R_1(C_1, C_2), R_5(C_2, C_3)$~~ 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) ~~$R_3(C_1, C_3) \leftarrow R_4(C_2, C_1), R_1(C_1, C_2)$~~ 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. 27 and Tab. 28, respectively. The results in Tab. 27 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 27: 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 28: 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. 29

Table 29: 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