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 <u>*point-wise*</u> *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 <u>point-wise</u> approaches, which perform reasoning w.r.t. <u>a single piece of clue</u> (e.g., a triple [1], a multi-hop query [20], a complex query graph [17]). For example in fact checking, given <u>a claim</u> (e.g., represented as a triple of the knowledge graph), it decides whether the claim is authentic or falsified [26, 2]. However, <u>comparative reasoning</u> ([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.

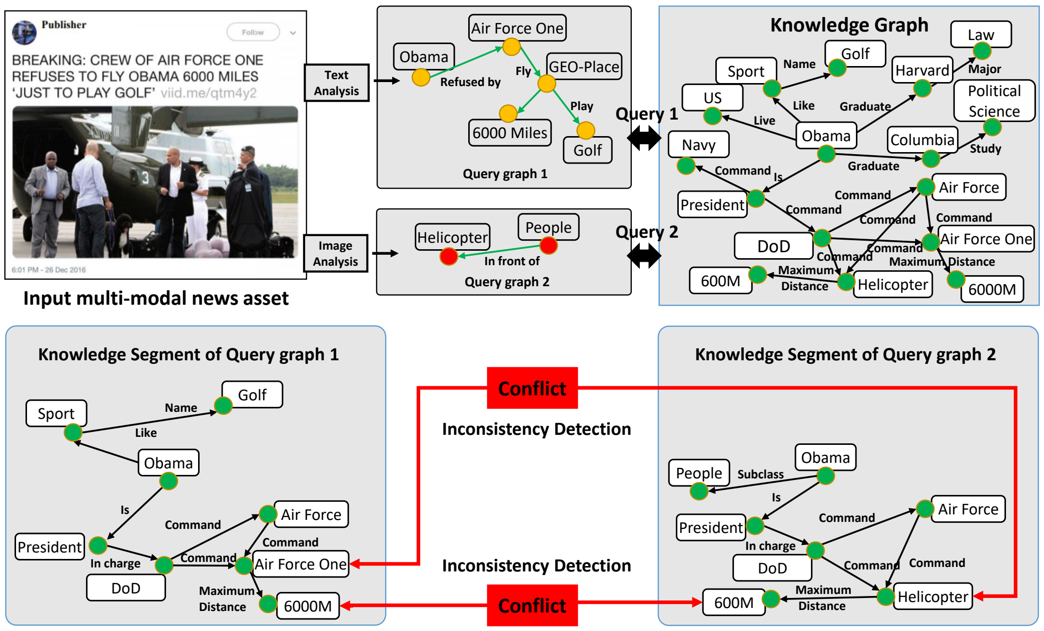
The main contributions of the paper are

- **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.


Publisher

BREAKING: CREW OF AIR FORCE ONE REFUSES TO FLY OBAMA 6000 MILES 'JUST TO PLAY GOLF' viid.me/qtm4y2

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], $\underline{K}$-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 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 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 $U(i, j) =$ TF$(i, j) \times IDF(j)$. Finally, the similarity of two predicates is defined as Sim(i, j) $=$ 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 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 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 $\underline{k}$-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 $< \texttt{s}_1, \texttt{p}_1, \texttt{o}_1 >$ and $< \texttt{s}_2, \texttt{p}_2, \texttt{o}_2 >$, we can divide it into the following six cases, including

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

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

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

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

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

C6. $\texttt{o}_1 = \texttt{s}_2$. For this case, no matter what $\texttt{p}_1$ and $\texttt{p}_2$ 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 $\texttt{p}_1$ contradicts $\texttt{p}_2$. For example, $\texttt{p}_1 = \texttt{isFather}$ and $\texttt{p}_2 = \texttt{isSon}$, 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.

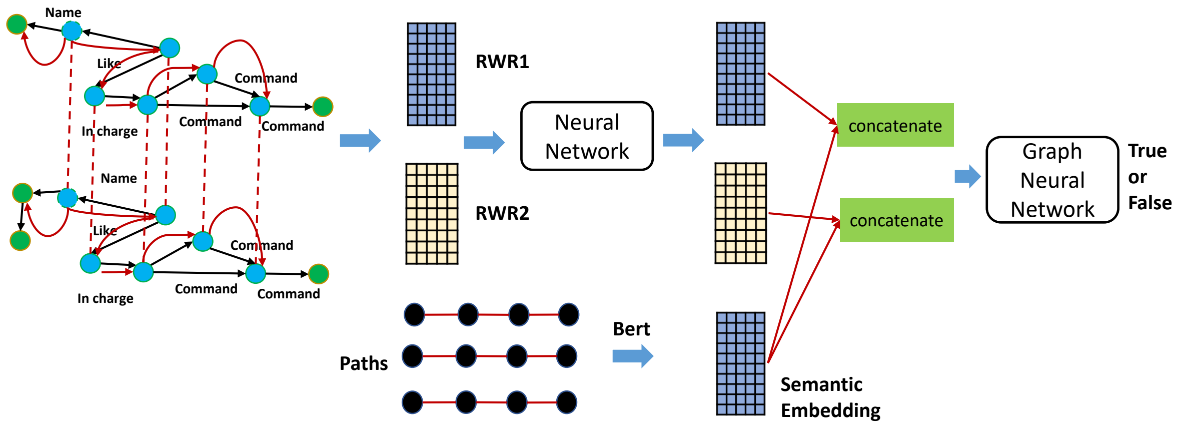


Figure 2: Neural network based pairwise comparative reasoning framework.

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

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

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

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

```
1: Input: Knowledge segment $KS_1$ and knowledge segment $KS_2$, pretrained embedding of all entities $\mathbf{E}$ in
   the knowledge graph.
2: Training:
3: Calculate random walks with restart matrices $\text{RWR}_1$ and $\text{RWR}_2$ for $KS_1$ and $KS_2$, respectively.
4: $E_1$ = NeuralNetwork($\text{RWR}_1$)
5: $E_2$ = NeuralNetwork($\text{RWR}_2$)
6: Get all node embedding in $KS_1$: $N_1$ = $\mathbf{E}(KS_1)$
7: Get all node embedding in $KS_2$: $N_2$ = $\mathbf{E}(KS_2)$
8: Concatenate embedding $C_1 = (E_1|N_1)$
9: Concatenate embedding $C_2 = (E_2|N_2)$
10: 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 $< \mathtt{o}_1$, `isTypeOf`, $\mathtt{o}_2 >$ and $< \mathtt{o}_2$, `isTypeOf`, $\mathtt{o}_1 >$. 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 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 Sim(KS_1, KS_2) in Eq. (9). Let $Q = (I - cN_\times A_\times)^{-1}$ and $S^{j,i}$ is a single entry matrix defined in Table 4. We have that

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

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

(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 $< \texttt{s}_1, \texttt{p}_1, \texttt{o}_1 >$ and $< \texttt{s}_1, \texttt{p}_2, \texttt{o}_2 >$, we need to decide whether $\texttt{o}_1$ belongs to $\texttt{o}_2$ or $\texttt{o}_2$ belongs to $\texttt{o}_1$. To this end, we build two new queries $< \texttt{o}_1$, `isTypeOf`, $\texttt{o}_2 >$ and $< \texttt{o}_2$, `isTypeOf`, $\texttt{o}_1 >$. 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 $< \texttt{o}_1$, `isTypeOf`, $\texttt{o}_2 >$ and $< \texttt{o}_2$, `isTypeOf`, $\texttt{o}_1 >$, 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:

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

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$(\texttt{isTypeOf}, e_i)$. If $\max\{\text{infTrans}(\texttt{o}_1, \texttt{o}_2), \text{infTrans}(\texttt{o}_2, \texttt{o}_1)\}$ is larger than a threshold T, then we say $\texttt{o}_1$ belongs to $\texttt{o}_2$ or $\texttt{o}_2$ belongs to $\texttt{o}_1$. 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.

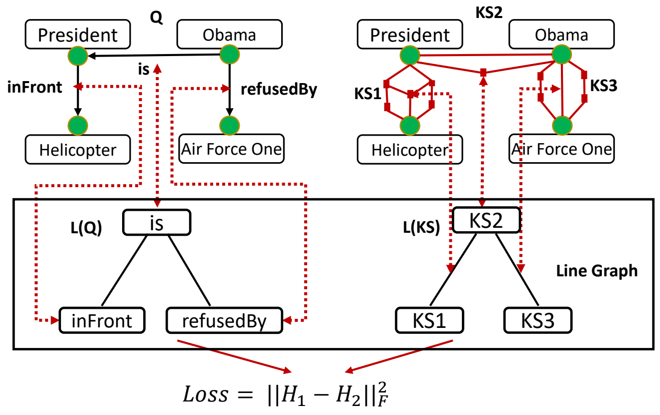


Figure 3: Collective comparative reasoning workflow.

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 $\text{Loss} = ||H_1 - H_2||_F^2$, 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 $< \mathtt{s}_1$, `isTypeOf`, $\mathtt{s}_2 >$ or $< \mathtt{s}_2$, `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 $< \mathtt{o}_1$, `isTypeOf`, $\mathtt{o}_2 >$ or $< \mathtt{o}_2$, `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., $< \mathtt{s}_1$, `isTypeOf`, $\mathtt{s}_2 >$, $< \mathtt{s}_2$, `isTypeOf`, $\mathtt{s}_1 >$, $< \mathtt{o}_1$, `isTypeOf`, $\mathtt{o}_2 >$ and $< \mathtt{o}_2$, `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.

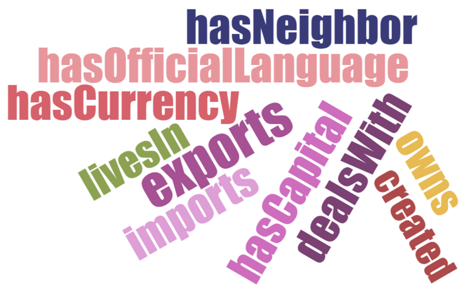
All the experiments are conducted on a moderate desktop with an Intel Core-i7 3.00GHz CPU and 64GB memory. The source code could be found at `https://github.com/lihuiliullh/KompaRe`. For TransE [1] in the experiments, we set the embedding dimension to 64 and use a margin of one and a learning rate of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

5.1 Predicate-Predicate Similarity Efficacy

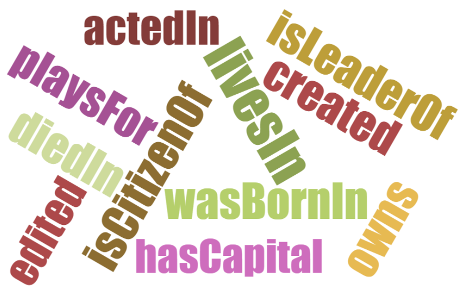
We evaluate the proposed predicate-predicate similarity. Figure 4 presents two examples on Yago dataset. It shows the top-$\underline{10}$ most similar predicates with `exports` and `livesIn`, respectively. The font size in Figure 4

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

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



(a) exports



(b) livesIn

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

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

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

Table 5: Predicate similarity of `isTypeOf` with others

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

5.2 Pair-wise Comparative Reasoning

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

Table 6: Accuracy of pair-wise comparative reasoning.

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

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

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

5.3 Neural Network Based Pair-wise Comparative Reasoning

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

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

Dataset / Model	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 ± **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 $< \texttt{s}_1, \texttt{p}_1, \texttt{o}_1 >$, $< \texttt{s}_1, \texttt{is}, \texttt{s}_2 >$ and $< \texttt{s}_2, \texttt{p}_2, \texttt{o}_2 >$, 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 $< \texttt{o}_1, \texttt{isTypeOf}, \texttt{o}_2 >$ and $< \texttt{o}_2, \texttt{isTypeOf}, \texttt{o}_1 >$, 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.

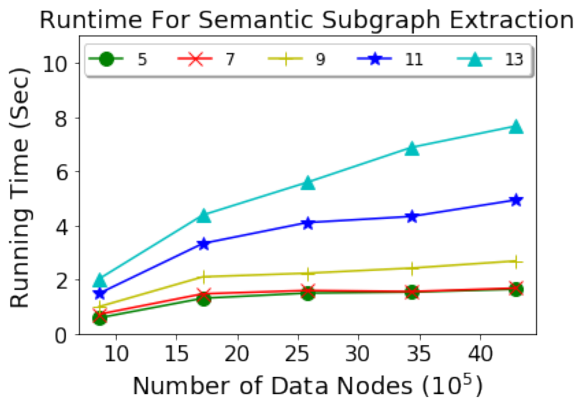
Dataset	# of queries	TransE	Jaccard	KL	KGMiner	Kompare
Positive	36	0.667	0.611	**1.000**	0.694	**1.000**
Negative	36	0.528	0.361	0.722	0.553	**0.863**
Average accuracy	-	0.598 ± 0.071	0.486 ± 0.126	0.861 ± 0.138	0.623 ± 0.071	**0.932 ± 0.063**

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.

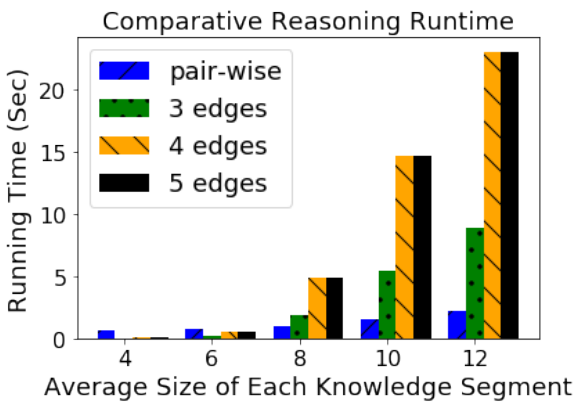
5.5 Efficiency Results

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.