Fact-Tree Reasoning for N-ary Question Answering over Knowledge Graphs

Yao Zhang[1] Peiyao Li[1] Hongru Liang[2] Adam Jatowt[3] Zhenglu Yang[1*]

[1]TKLNDST, CS, Nankai University, China, [2]Sichuan University, China,
[3]University of Innsbruck, Austria

{yaozhang, peiyao_li}@mail.nankai.edu.cn, lianghongru@scu.edu.cn,
adam.jatowt@uibk.ac.at, yangzl@nankai.edu.cn

Abstract

Current Question Answering over Knowledge Graphs (KGQA) task mainly focuses on performing answer reasoning upon KGs with binary facts. However, it neglects the n-ary facts, which contain more than two entities. In this work, we highlight a more challenging but under-explored task: n-ary KGQA, i.e., answering n-ary facts questions upon n-ary KGs. Nevertheless, the multi-hop reasoning framework popular in binary KGQA task is not directly applicable on n-ary KGQA. We propose two feasible improvements: 1) upgrade the basic reasoning unit from entity or relation to fact, and 2) upgrade the reasoning structure from chain to tree. Therefore, we propose a novel fact-tree reasoning framework, FacTree, which integrates the above two upgrades. Fac-Tree transforms the question into a fact tree and performs iterative fact reasoning on the fact tree to infer the correct answer. Experimental results on the n-ary KGQA dataset we constructed and two binary KGQA benchmarks demonstrate the effectiveness of FacTree compared with state-of-the-art methods.

1 Introduction

The task of Question Answering over Knowledge Graphs (KGQA) has provided new avenues to the recent development of QA systems by utilizing the advantages of KGs (Yu et al., 2017; Dubey et al., 2019; Huang et al., 2019; Zhang et al., 2021). Current KGQA studies mainly consider performing answer reasoning upon KGs with binary facts, which encode binary relations between pairs of entities, e.g., _Golden State Warriors' arena is Chase Center_[1]. However, n-ary facts that involve more than two entities are also ubiquitous in reality (Guan et al., 2019; Abboud et al., 2020; Wang et al., 2021), e.g., the ternary fact _Golden State Warriors won_

---
*Corresponding author.
[1]Entities are underlined.

the NBA championship in 2018. Compared to binary facts, n-ary facts have more information content. This makes the answer reasoning for questions involving n-ary facts more intractable, exposing open challenges in KGQA. In this work, we aim to study the under-explored **n-ary KGQA** task, i.e., answering n-ary facts questions upon n-ary KGs.

The multi-hop reasoning KGQA method (Das et al., 2018; Qiu et al., 2020; Saxena et al., 2020; Ren et al., 2021) has become popular for its high efficiency and interpretability. Specifically, the reasoning process can be expressed as a chain, starting from an entity extracted from the question and then walking on the KG by connected relations and entities until arriving at the answer entity. See Figure 1 (b) for an example, to answer the question, _what is the address of the arena of the Golden State Warriors_, the reasoning chain starts from `Golden State Warriors`, to walk through "arena→Chase Center→address", and it ends at `1 Warriors Way`, i.e., the answer. Multi-hop reasoning has been studied widely on the binary KGQA task. Here, we first try to execute it on the n-ary KGQA task.

However, we find that multi-hop reasoning is not directly applicable on n-ary KGQA. We take the n-ary facts question in Figure 1 (c) as an example to explain. First, the essence of multi-hop reasoning is to construct a reasoning chain by treating the relation as the translation between two entities (Bordes et al., 2013; Ren and Leskovec, 2020), naturally in a linear structure. However, the transition from binary to n-ary facts is similar to the transition from a line to a plane. For a single n-ary fact (e.g., _Golden State Warriors won the NBA championship in 2018_), the reasoning chain could only include two entities `Golden State Warriors` and `NBA championship` and a relation `win` involving them, leading to the possible loss of important information `2018`. To overcome
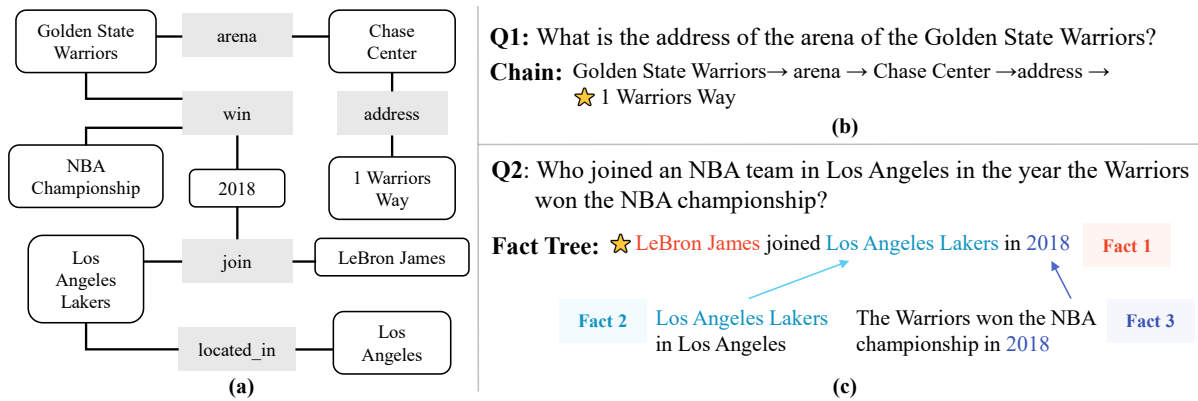
Figure 1: (a) A KG fragment, where entities are represented by round rectangles. `Win` and `join` are two ternary relations. (b) and (c) are two QA examples, where the correct answer is marked by a star. The multi-hop reasoning method can be used to answer Q1, and the reasoning process can be visualized as a chain (b). However, for the more complex Q2, the multi-hop reasoning method is not applicable. We use fact as the basic reasoning unit to construct the reasoning process. As shown in (c), the reasoning process can be visualized as a fact tree: **fact 2** and **fact 3** are leaf nodes need to be inferred first, and then the two inferred entities (`Los Angeles Lakers` and `2018`) are transmitted to the root node (**fact 1**). Finally the root node infers the answer entity (`LeBron James`).

this weakness, we propose that **upgrading the basic reasoning unit from an entity or relation to the fact** to expand the coverage of information during reasoning.

Nevertheless, multi-hop reasoning would still be less capable in more complex reasoning scenarios where a question involves multiple n-ary facts. For example, the question in Figure 1 (c) is composed of three facts. When using fact as the basic reasoning unit, the whole reasoning process can be represented as a tree structure, where nodes represent facts and edges reflect the reasoning order. Specifically, in the fact tree, the entities (`Los Angeles Lakers` and `2018`) which are missing in the two leaf nodes (fact 2 and fact 3) are first inferred and then passed to the root node (fact 1). The root node can then finally infer the correct answer entity `LeBron James`. Obviously, the chain structure used in the multi-hop reasoning framework is evidently insufficient to cope with the tree structure. Therefore, to improve the ability to cope with more complex reasoning scenarios, we propose that **upgrading the reasoning structure from chian to tree**.

In this work, we propose a novel **fact-tree reasoning framework**, namely, **FacTree**, which integrates the above two upgrades and pipelines the answer reasoning process into three steps: 1) fact tree construction, which transforms an input natural language (NL) question into an NL fact tree; 2) fact location, which locates the NL fact onto the KG; and 3) fact reasoning, which iterates intra-fact

and inter-fact reasoning to infer the answer. During the intra-fact reasoning, the n-ary KG embedding model (Guan et al., 2020) is plugged in to alleviate the deficiency of KG incompleteness. The explicit tree reasoning structure makes the results strongly interpretable. Furthermore, we develop a new dataset called WikiPeopleQA to foster research on n-ary KGQA. We then conduct comprehensive experiments on WikiPeopleQA dataset to show that FacTree has the desired ability to perform effective reasoning on n-ary fact questions. Besides, on two binary KGQA datasets, FacTree also indicates a strong ability to infer answers compared with state-of-the-art methods accurately.

Our study fundamentally contributes to bridging the gap between binary KGQA and n-ary KGQA. The proposed FacTree can serve as a preliminary foundation for the n-ary KGQA. To summarize, our contributions are:

- We highlight a more challenging task: n-ary KGQA than a standard binary KGQA task setting. We further observe that the multi-hop reasoning framework popular in binary KGQA is no longer applicable to n-ary KGQA.

- We propose a novel fact-tree reasoning framework, FacTree, which can serve as a preliminary foundation for n-ary KGQA study. And we develop a new dataset: WikiPeopleQA to foster research on n-ary KGQA.

- We conduct comprehensive experiments to show that our framework has the desired reasoning ability for both n-ary and binary KGQA tasks.

789

Q: Who joined an NBA team in Los Angeles in the year the Warriors won the NBA championship? **A**: LeBron James
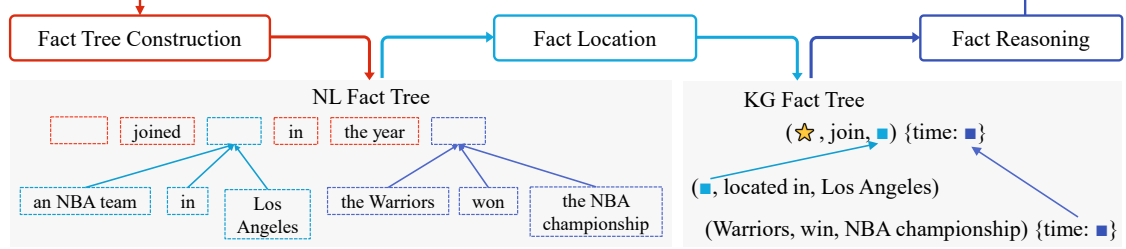

Figure 2: Overview of FacTree. It takes the question (**Q**) as an input, passes through a three-stage pipeline processing: 1) fact tree construction (Sec. 3.1), 2) fact location (Sec. 3.2) and 3) fact reasoning (Sec. 3.3), and finally gets the answer entity (**A**). Placeholders □ in the NL and KG fact trees indicate the entities to be inferred.

2 Related Work

The previous series of KGQA models (Liang et al., 2011; Berant et al., 2013; Yih et al., 2014; Lan and Jiang, 2020; Sun et al., 2020; Wolfson et al., 2020) synthesize a structured query graph from the question and then match the query with KG to get the answer. This type of model has high interpretability but is challenged by the incomplete nature of KGs. Then another series models compute the semantic similarity of the question and each candidate answer directly in the latent space (Bordes et al., 2015; Dong et al., 2015; Hamilton et al., 2018; Zhang et al., 2018b). This type of model overcomes the limitation of incomplete KG, but lacks sufficient interpretability. FacTree uses facts as the basic reasoning unit to alleviate the deficiency of KG incompleteness and the explicit tree reasoning structure to realize strongly interpretable.

Multi-hop reasoning framework has attracted widespread attention due to its high flexibility and high interpretability in recent years (Fu et al., 2020). Current efforts build an explicit reasoning chain through training a reinforcement learning agent to walk on the KG (Das et al., 2018; Qiu et al., 2020; Kaiser et al., 2021), or construct implicit reasoning chains through memory network (Sukhbaatar et al., 2015; Miller et al., 2016; Chen et al., 2019) or in the latent space (Bordes et al., 2014; Saxena et al., 2020; Ren and Leskovec, 2020; He et al., 2021; Ren et al., 2021). This kind of method performs well on binary fact questions but has difficulties in dealing with n-ary fact questions. Of course, one could construct and synthesize multiple reasoning chains to tackle n-ary KGQA. But this would inevitably lead to an exponential increase in the reasoning difficulty and computational complexity, which in turn affects the reasoning performance.

Our work first highlights the n-ary KGQA task.

The research of n-ary KG provides a feasible research foundation for n-ary KGQA. KG embedding learning on n-ary facts (Wen et al., 2016; Zhang et al., 2018a; Fatemi et al., 2019; Guan et al., 2019, 2020; Abboud et al., 2020) has grown considerably in recent years. The n-ary KG embedding model (Guan et al., 2020) is plugged in FacTree to alleviate the reasoning difficulties caused by the KG incompleteness.

3 Fact-tree Reasoning

We illustrate the fact-tree reasoning framework for n-ary KGQA in Figure 2. It takes the question (**Q**) as an input, passes through a three-stage pipeline processing, and finally gets the answer entity (**A**). In the first fact tree construction stage, we construct the NL fact tree from the NL question (Sec. 3.1). In the second fact location stage, we locate the NL fact onto the KG to obtain the KG fact tree (Sec. 3.2). This helps to bridge the semantic and structure gap between the unstructured NL and the structured KG. In the last fact reasoning stage, we perform intra-fact and inter-fact reasoning iteratively on the KG fact tree to infer the answer entity (Sec. 3.3).

We use fact as the basic reasoning unit in the fact-tree reasoning framework. In the NL fact tree, the fact is represented as a sequence of words and placeholders, where the words are taken from the NL question, and the placeholders refer to the missing entities to be inferred, e.g., the fact (□ *joined* □ *in the year* □). In the KG fact tree, following (Guan et al., 2020), we represent the n-ary fact as

$$fact = (s, p, o), \{a_1 : v_1, a_2 : v_2, ..., a_m : v_m\}, \quad (1)$$

where (s, p, o) denotes the subject-predicate-object information in the fact, named primary triple; each $a_i : v_i$ ($i \in \{1, 2, ..., m\}$) is an attribute value pair, a.k.a., the auxiliary description to
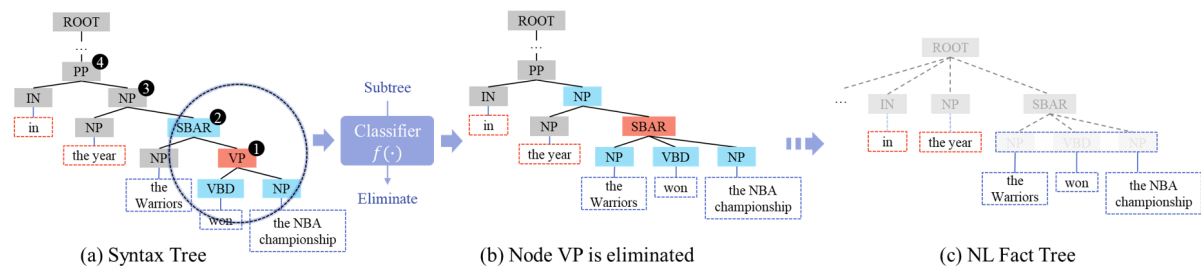
Figure 3: An example of NL fact tree construction (partial steps). The eliminations of nodes in the order of ❶→❷→❸→❹→ · · · ·. Best viewed in color.

the primary triple. For example, the ternary fact *LeBron James joined Los Angeles Lakers in 2018* is formalized as (LeBron James, join, Los Angeles Lakers),{time: 2018}. Note that the binary fact only contains the primary triple.

3.1 Fact tree construction

In FacTree, we use fact as the basic reasoning unit and use the tree structure to represent the associations among facts. Here, we design an automatic NL fact tree construction algorithm to transfer the NL question to the NL fact tree. Since the syntax tree naturally expresses the hierarchical relation of the elements of the sentence and in order to facilitate the subsequent locating of the NL facts into KG, we use the syntax tree[2] of Q as the initial structure. We expect the constructed NL fact tree to satisfy the following characteristics: 1) the leaf nodes are words or phrases of Q; and 2) if the leaf nodes share the same parent node, they belong to the same fact. Therefore, the NL fact tree construction algorithm can be viewed as an iterative eliminating of nodes in the syntax tree to achieve clustering of nodes within facts and differentiation between facts.

The node elimination process starts from the antepenult level of the syntax tree and proceeds from bottom to top. We observe that two semantically different questions may be parsed into the same syntax structure except for the leaf nodes. Therefore, disregarding leaf nodes makes our algorithm more adaptable. Also, the parent node of a leaf node needs to be reserved for identifying the leaf node. Figure 3 shows a specific example [3]. As shown in

---
[2]We use the Stanford Parser to generate the syntax tree. The leaf nodes are words or phrases of Q and the branch nodes are syntax labels, e.g., NP (Noun Phrase) and VP (Verb Phrase).

[3]Due to the space limitation, only part of the syntax tree and the corresponding elimination steps are shown here.

Algorithm 1: NL Fact Tree Construction

Input: The question Q, empty node stacks $\mathbb{V}, \mathbb{V}'$;
Output: The NL fact tree FT;

1 **Initialization**
2   FT = Parse(Q),
3      $\mathbb{V}$ =BFS(FT);
4 **while** $\mathbb{V}$ **do**
5      $v = \mathbb{V}.pop()$;
6   **if** $v.isLeaf()$ **or** $v.children.isLeaf()$ **then**
7   continue;
8   **else**
9          $\mathbb{V}'.push(v)$;
10   **end**
11 **end**
12 **while** $v \neq$ FT.root **do**
13      $v = \mathbb{V}'.pop()$;
14      $T_v = \{v\} \cup \{v.parent\} \cup \{v.children\}$;
15   **if** $f(T_v) ==$ *eliminate* **then**
16   update
17   **for** each *child* in $v.children$ **do**
18              $child.set\_parent(v.parent, \text{FT})$
19   **end**
20   FT.$delete(v)$
21   **end**
22 **end**

(a), the pruning starts from the node VP (❶, colored in red). To decide whether to eliminate this node or not, we extract a subtree that contains the node and its neighbor nodes (colored in blue). This subtree is fed into a classifier $f(\cdot)$, which is composed of a Graph Convolutional Network (GCN) as embedding layer and a fully-connected layer. If $f(\cdot)$ outputs "eliminate", the node will be eliminated and its children will be directly connected to its parent, as shown in (b). Otherwise, this node will be retained. This process continues until the iteration meets the root node. Finally, we remove non-leaf nodes and keep the hierarchical structure of leaf nodes. The nodes in the upper-layer facts that are connected to the lower-layer facts are replaced with placeholders, as shown in (c). We summarize the construction of NL fact tree in Algorithm 1.

Specifically, for the GCN, we use the propagation rule for calculating the node embedding update
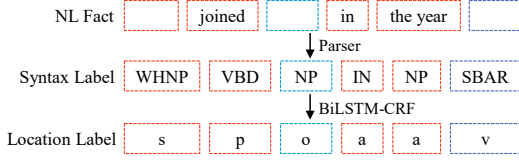
791

Figure 4: Example of a structure matching process.

Figure 5: Example of an iterative reasoning process. The order of reasoning is ❶→❷→❸→❹→❺.

for each layer as follows:

$$h_v^{(i+1)} = \sigma_i(h_v^{(i)} W_0^{(i)} + \sum_{u \in T_v \setminus v} h_u^{(i)} W_1^{(i)}), \quad (2)$$

where v and T_v follow the definitions in Algorithm 1, u is one of the neighbors of v, $h_*^{(i)}$ represents the hidden layer activations of nodes in the ith layer, $\sigma_i(\cdot)$ is the activation function, and $W_{0,1}^{(i)}$ are the i-layer weight matrices.

3.2 Fact Location

This stage aims to transfer the NL fact tree to the KG fact tree, specifically, to locate each NL fact in the tree to a KG fact. It is divided into three specific steps: 1) entity linking, 2) structure matching, and 3) relation extraction.

During entity linking, following the standard setting in KGQA (Saxena et al., 2020), we assume that the entities of the question are given and linked to nodes on the KG. Note that the placeholders are directly reserved and they indicate the entities that need to be inferred.

The key of structure matching is to locate the subject s, predicate p, object o, attribute a and value v in the NL fact. We view this process as a sequence labelling task, as shown in Figure 4. We believe that location labels are more strongly associated with syntax labels, compared to word sequences. Therefore, the input is the sequence formed by the syntax labels of each node in the NL fact. The output is the sequence of location labels, and the label set is $\{s, p, o, a, v\}$. Here, we adopt the BiLSTM-CRF model (Huang et al., 2015) to perform sequence labelling.

After entity matching, we conduct relation extraction, i.e., transferring the word sequences labelled p or a to the corresponding relations (predicates or attributes) in KG. Specifically, we adopt pre-trained SBERT model (Reimers and Gurevych, 2019) to get the embeddings of the relation in KG (i.e., r) and the word sequences (i.e., w). Then we use cosine similarity as a scoring function $s(\cdot)$ to assign scores to the two embeddings and select
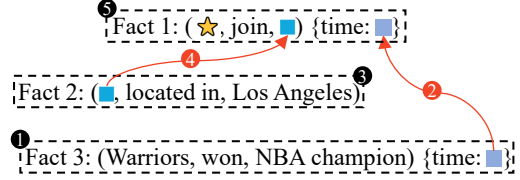
the relation with the highest score:

$$s(r, w) = \frac{r \cdot w}{\|r\|\|w\|} = \frac{\sum_{i=1}^n r_i w_i}{\sqrt{\sum_{i=1}^n r_i^2}\sqrt{\sum_{i=1}^n w_i^2}}. \quad (3)$$

Finally, combining the predicted location labels, and linked entities and extracted relations, NL facts can be transformed into KG facts (cf. Figure 2). Placeholders are the bridge between upper-layer and lower-layer facts. Interestingly, due to the variability of NL organization, there may be no placeholder in the lower-layer fact. It is because when constructing the NL fact tree, the placeholder (usually value) is assigned to upper-layer fact according to the syntax structure. Therefore, we directly copy the upper-layer placeholder directly to the lower-layer fact. For example, in Figure 2, the fact *the Warriors won the NBA championship* will be transformed to (Golden State Warriors, win, NBA championship),{time:2018}, where the attribute time is copied from the upper-layer fact.

3.3 Fact Reasoning

In this stage, we perform the inter-fact and intra-fact reasoning iteratively based on the KG fact tree to find the answer entity. One example of iterative reasoning process is shown in Figure 5.

Inter-fact Reasoning The whole process of inter-fact reasoning is carried out in a bottom-up manner. Specifically, the entity inferred from the lower-layer fact will be transferred to the upper-layer fact. For example, the entity 2018 inferred from fact 3 will be transferred to fact 1, i.e., the second step in Figure 5.

Intra-fact reasoning This module aims to infer the missing entity of each incomplete fact. We formulate this process as the KG completion task. KG embedding models (Bordes et al., 2013; Dettmers et al., 2018; Guan et al., 2019, 2020) are studied to deal with this task, by learning entity and relation embeddings and designing a scoring function to infer the missing entity. In this work, we use

Table 1: Statistics of WikiPeopleQA, WC2014 and PathQuestion, as well as their subsets. Note that the n-ary here means that n is greater than 2.

| Dataset | # Question-Answer Pair | | Background KG | | | |
	Total	Subset	# Fact	# Entity	# Relation(binary)	# Relation(n-ary)
WikiPeopleQA (1F/2F/3F)	4,491	2,365 / 1,497 / 629	56,426	28,043	150	557
WC2014 (1H/2H/C)	10,162	6,482 / 1,472 / 2,208	6,482	1,127	6	0
PathQuestion (2H/3H)	7,106	1,908 / 5,198	4,049	2,215	14	0

NeuInfer (Guan et al., 2020), a KG embedding model that can perform on binary and n-ary facts, to implement intra-fact reasoning.

However, comparing with the traditional KG completion task, the missing entity needs not only to complete the current fact, but also to satisfy the upper-layer fact. For example, in fact 2, `Sunset Boulevard` and `Los Angeles Lakers` are all located in Los Angeles. While, considering the upper-layer fact 1, the missing entity needs to satisfy the fact that the predicate is `join`. Therefore, we introduce a score amplification mechanism: if an alternative entity can satisfy the upper-layer fact, its corresponding score will be magnified λ times.

3.4 Training

The classifier $f(\cdot)$ and BiLSTM-CRF model are trained in a supervised manner, where the training signals are obtained from manually labeled (syntax tree, NL fact tree) and (syntax label sequence, location label sequence) pairs, respectively. We observe the syntax structure of different questions may be similar or even consistent. Therefore, we reduce the the input space by using syntax-related information rather than NL, to relieve the manual annotation pressure and also the learning difficult.

4 Experiments

4.1 Dataset

In this work, we target at studying the n-ary KGQA task. Considering the popular KGQA datasets involve almost exclusively binary facts, we develop an n-ary KGQA dataset: WikiPeopleQA (abbr. WP), in which questions involve multiple n-ary facts and the background KG is also composed of n-ary facts. We also conduct evaluation on two binary KGQA benchmarks: WC2014 (abbr. WC) (Zhang et al., 2016) and PathQuestion (abbr. PQ) (Zhou et al., 2018). Depending on the number of **F**acts or

Hops involved in the question[4], WikiPeopleQA is divided into WP-1F, WP-2F and WP-3F. WC2014 is divided into WC-1H and WC-2H, as well as a conjunctive question set WC-C. PathQuestion is divided into PQ-2H and PQ-3H. We partition the three datasets into train/valid/test subsets with a proportion of $8 : 1 : 1$. The detailed statistics are shown in Table 1.

4.2 Experimental Setup

4.2.1 Training Details

During training the classifier, the embedding size of node is 50, the learning rate is $1e-5$, and the number of GCN layers is 3. During training the BiLSTM-CRF model, the embedding size of node is 100, the learning rate is $2e-5$. We use Adam optimizer for optimization for above training process. The hyper-parameter λ is set to 1.5. The training process of the KG embedding model used in the fact reasoning stage is following (Guan et al., 2020). The experimental results are all averaged across three training repetitions.

4.2.2 Baselines

We compare our framework with a series of popular multi-hop reasoning baseline approaches. According to the form of the reasoning chain, these baselines can be divided into two categories. The first category baseline builds an explicit reasoning chain through training an agent to walk on the KG: IRN-weak (Zhou et al., 2018), MINERVA (Das et al., 2018) and SRN (Qiu et al., 2020); the second category baseline builds an implicit reasoning chain through memory network or in the latent space: MemNN (Sukhbaatar et al., 2015), KV-MemNN (Miller et al., 2016) and Embed-KGQA (Saxena et al., 2020). We also compare with QGG (Lan and Jiang, 2020), which synthesizes a query graph from the question and then match the query with KG to get the answer. When

---

[4]Note that, for a binary fact question, the hop number is generally equal to the fact number.

Model	WP	WP-1F	WP-2F	WP-3F	WC	WC-1H	WC-2H	WC-C	PQ	PQ-2H	PQ-3H
MemNN (Sukhbaatar et al., 2015)	<u>32.9</u>	34.2	39.6	12.5	52.4	71.6	55.5	73.3	86.8	89.5	79.2
KV-MemNN (Miller et al., 2016)	24.5	15.0	<u>40.0</u>	<u>16.0</u>	76.7	87.0	87.0	78.8	85.2	91.5	79.4
EmbedKGQA (Saxena et al., 2020)	26.4	35.4	22.3	3.5	52.5	59.6	79.0	52.0	36.7	51.0	30.6
IRN-weak (Zhou et al., 2018)	-	-	-	-	78.6	83.4	92.1	83.7	85.8	91.9	83.3
MINERVA (Das et al., 2018)	10.9	20.5	0.3	0.2	89.6	87.2	93.1	82.4	73.1	75.9	71.2
SRN (Qiu et al., 2020)	13.3	24.9	0.3	0.8	<u>96.5</u>	<u>98.9</u>	**97.8**	<u>87.3</u>	<u>89.3</u>	<u>96.3</u>	<u>89.2</u>
QGG (Lan and Jiang, 2020)	24.9	<u>41.8</u>	1.1	0.0	94.0	94.9	92.9	**99.9**	40.4	67.8	28.9
FacTree	**54.4**	**63.1**	**47.0**	**40.1**	**99.5**	**99.9**	<u>96.3</u>	**99.9**	**92.8**	**98.4**	**90.8**

Table 2: Model Performance on n-ary KGQA task (WikiPeopleQA dataset) and binary KGQA task (WC2014 and PathQuestion datasets) under the accuracy(%) metric (pairwise t-test at 5% significance level). The best performance results are shown in **bold**, and the second best results are shown in <u>underlined</u>.

evaluating explicit multi-hop reasoning methods and QGG, we use the dummy entity to divide n-ary facts in the KG of WikiPeopleQA into binary facts.

4.3 Main Results

Performance of FacTree Table 2 presents the statistics of model's performances both on the n-ary and binary KGQA tasks. We can see that Fac-Tree achieves significantly higher accuracy than state-of-the-art baselines on the n-ary KGQA task. Specifically, compared with the best performing multi-hop reasoning baseline MemNN, FacTree improves accuracy by 21.5% (w.r.t., WP). We have following discoveries:

- FacTree shows large advantages on coping with complex questions with multiple facts. On the WP-3F sub-dataset, comparing with baselines, FacTree has made a qualitative leap on accuracy (16.0→40.1). This confirms that multi-hop reasoning methods are inapplicable to more complex reasoning scenarios.

- Interestingly, the explicit multi-hop reasoning methods (e.g., SRN) are obviously weaker than implicit methods (e.g., EmbedKGQA). This is because the n-ary facts are split by dummy entities, adding difficulty to build explicit reasoning chains. Implicit methods weaken the distinction between binary and n-ary facts, which makes it more flexible in dealing with n-ary facts.

- The QGG method directly matches the generated query graph to the KG, resulting in performing well on questions with less facts, but clearly lacking flexibility for question with more facts.

Besides, fact-tree reasoning also achieves a good performance on binary KGQA. We observe a large performance gap between multi-hop reasoning
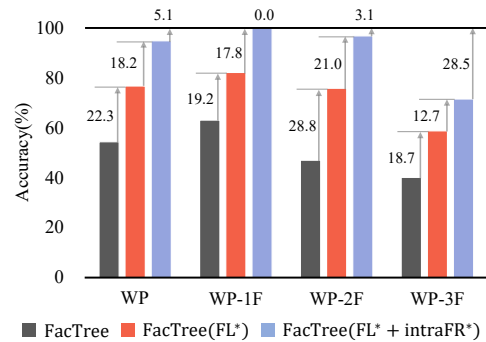

Figure 6: Staged evaluation of FacTree. FL^* and $intraFR^*$ denote turning off the FL and intraFR components respectively. Note that the accuracy of FacTree(FL^*+$intraFR^*$+$interFR^*$) is 100%.

methods on binary and n-ary KGQA tasks. Therefore, it would be valuable to pay more attention to the study of n-ary KGQA.

Staged Evaluation We test the capability of each stage of FacTree. Because the fact-tree reasoning framework is a pipeline structure, the reasoning error always occurs in cascade. We turn off the components on the pipeline from the beginning to see the impact on the overall effect. "Turn off a component" means to replace the real output with the ground truth, that is, the component is perfect by default. We test three components of FacTree: fact location (FL), intra-fact reasoning (intraFR) and inter-fact reasoning (interFR). The error caused by the fact tree construction stage is negligible here because of the relatively small number of NL fact tree types in the datasets. Figure 6 shows turning off the components in turn will lead to an accuracy increase of 22.3% (FL), 18.2% (intraFR) and 5.1% (interFR) respectively (w.r.t., WP). Impressively, transferring the NL fact tree to KG fact tree and inferring missing entities of incomplete facts are

Stage	Question	Error Instance	Note
Fact Location	Which political party does William's father join in?	who is the mother of [ ] → (⭐, **sibling**, ■) / whose father is William I Count of Nassau Dillenburg → (■, father, William I Count of Nassau Dillenburg)	The true relation is <u>member of political party</u>.
Intra-Fact Reasoning	Who is sibling of Liu Yan?	(⭐, sibling, Liu Yan) → (**Liu Cang**, sibling, Liu Yan)	The true answer is <u>Liu Kang</u>.
Inter-Fact Reasoning	Who win award Best Female Athlete ESPY Award in the time when Dwight Howard win the NBA Defensive Player of the Year Award?	(⭐, win, Best Female Athlete ESPY Award) {time: ■} / (Dwight Howard, win, NBA Defensive Player) {time: ■} → (⭐, win, Best Female Athlete ESPY Award) {time: **2019**} / (Dwight Howard, win, NBA Defensive Player) {time: **2019**}	<u>2019</u> satisfies the lower-layer fact, but the upper-layer fact needs <u>2010</u> or <u>2011</u>.

Figure 7: Error instances in the fact location, intra-fact and inter-fact reasoning stages, respectively.

the two keys that affect the reasoning accuracy. Moreover, the influence of the inter-fact reasoning module becomes apparent as the number of facts increases. This is because it may happen that an entity satisfying the lower-layer fact may not be able to satisfy the upper-layer fact. For instances of errors in each module, please see Section 4.4.

4.4 Analysis of FacTree

Performance w.r.t. Incomplete KGs As the capacity of the KG continues to expand, current KGs are typically incomplete with many facts missing. Incomplete KGs put forward higher requirements on the capabilities of KGQA models. Therefore, we conduct an experiment on FacTree and two popular baselines to test their reasoning capability for incomplete KGs. As shown in Table 3, when the KG is reduced by half, the effect of our model decreases the least. This is because we adopt KG embedding models to perform intra-fact reasoning in FacTree. This design relaxes the requirements for KG completeness. SRN requires the construction of query graph or explicit reasoning chain on KG, so it is more sensitive to the incompleteness of KG. EmbedKGQA also uses KG embedding models. The performance gap between it and our FacTree corroborates the superiority of the fact-tree reasoning framework.

Zero-shot Learning w.r.t. Classifier $f(\cdot)$ We conduct a zero-shot learning experiment to test the capability of our proposed classifier $f(\cdot)$ in the fact construction stage though a 5-fold cross validation. We evaluate the accuracy of the constructed NL fact tree on the mixed dataset combined with three datasets (WikiPeopleQA, WC2014 and PathQuestion). The mixed dataset is divided into five parts according to the NL fact tree classes. For each

Model	WP	WP-50%	WC	WC-50%
SRN	13.3	0.1 (↓99%)	96.5	0.0 (↓100%)
EmbedKGQA	26.4	6.5 (↓75%)	52.5	11.0 (↓79%)
FacTree	54.4	17.8 (↓67%)	99.5	37.2 (↓63%)

Table 3: Performance on incomplete KGs. ↓ indicates the decrease in accuracy when the KG is halved.

Size	O	O+F	O+C	O+F+C	O+F+C+S
Acc.	52.9	53.9	63.1	91.4	91.0

Table 4: Performance of different subtree range w.r.t. fact tree construction stage. "F", "C" and "S" denote the father, child and sibling nodes of the central node "O", respectively.

fold, the fact tree classes in the testing set do not appear in the training set. Based on this setting, our classifier can reach 81.2% accuracy with a standard deviation of 0.098. This indicates our classifier has the scalable ability to construct unseen fact trees.

Effectiveness of GCN We also test the effect of the execution range of the GCN on the accuracy of the fact tree construction. The range of GCN execution is a subtree of the syntax tree containing the central node to be eliminated and its neighbor nodes. A total of five range types are tested depending on whether the father, siblings and child of the central node were included. As shown in Table 4, the optimal subtree range includes the central node and its father and child nodes. Interestingly, the addition of sibling nodes did not bring significant effect improvement.

Error Analysis We conduct a qualitative study on error instances, as shown in Figure 7, and analysis the directions for future work. In the fact location

stage, introducing more effective relation extraction techniques can contribute to reduce relation extraction error. In the intra-fact reasoning stage, it is necessary to improve KG embedding model's capability. There is a false negative error case. For example, the fact (`Liu Cang, sibling, Liu Yan`) is true, but is not included by KG, resulting in the inferred answer being judged as wrong. So, adopting broader KGs is suggested in future studies. In the inter-fact reasoning stage, entities can be incompatible when transferred between facts. Therefore, intra- and inter-factual reasoning needs to act more closely together to reduce the incompatibility.

5 Conclusion

This work highlights a more challenging task: n-ary KGQA, and it advocates that the multi-hop reasoning framework popular in binary KGQA is no longer applicable to n-ary KGQA. A novel fact-tree reasoning framework FacTree is proposed, which pipelines the n-ary KGQA into three steps: fact tree construction, fact location, and fact reasoning to infer the correct answer. The quantitative and qualitative experimental results have demonstrated that FacTree has superior reasoning ability on n-ary and binary fact questions.

6 Acknowledgements

References

Ralph Abboud, İsmail İlkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. 2020. Boxe: A box embedding model for knowledge base completion. In *Proceedings of NeurIPS*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, pages 1533–1544.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of EMNLP*, pages 615–620.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. In *arXiv preprint arXiv:1506.02075*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NeurIPS*, pages 2787–2795.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2019. Bidirectional attentive memory networks for question answering over knowledge bases. In *Proceedings of NAACL-HLT*, pages 2913–2923.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *Proceedings of ICLR*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of AAAI*, pages 1811–1818.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over Freebase with multi-column convolutional neural networks. In *Proceedings of ACL-IJCNLP*, pages 260–269.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *Proceedings of ISWC*, pages 69–78.

Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. 2019. Knowledge hypergraphs: Prediction beyond binary relations. In *arXiv preprint arXiv:1906.00137*.

Bin Fu, Yunqi Qiu, Chengguang Tang, Yang Li, Haiyang Yu, and Jian Sun. 2020. A survey on complex question answering over knowledge base: Recent advances and challenges. In *arXiv preprint arXiv:2007.13069*.

Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 2020. NeuInfer: Knowledge inference on N-ary facts. In *Proceedings of ACL*, pages 6141–6151.

Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2019. Link prediction on n-ary relational data. In *Proceedings of WWW*, pages 583–593.

William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. In *Proceedings of NeurIPS*, pages 2030–2041.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of WSDM*, pages 553–561.

Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *Proceedings of WSDM*, pages 105–113.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. In *arXiv preprint arXiv:1508.01991*.

Magdalena Kaiser, Rishiraj Saha Roy, and Gerhard Weikum. 2021. Reinforcement learning from reformulations in conversational question answering over knowledge graphs. In *Proceedings of SIGIR*.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of ACL*, pages 969–974.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of ACL*, pages 590–599.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of EMNLP*, pages 1400–1409.

Yunqi Qiu, Yuanzhuo Wang, Xiaolong Jin, and Kun Zhang. 2020. Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision. In *Proceedings of WSDM*, pages 474–482.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of EMNLP*, pages 3982–3992.

Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. 2021. Lego: Latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *Proceedings of ICML*, pages 8959–8970.

Hongyu Ren and Jure Leskovec. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs. In *Proceedings of NeurIPS*.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of ACL*, pages 4498–4507.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of NeurIPS*, pages 2440–2448.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. Sparqa: Skeleton-based semantic parsing for complex questions over knowledge bases. In *Proceedings of AAAI*, pages 8952–8959.

Quan Wang, Haifeng Wang, Yajuan Lyu, and Yong Zhu. 2021. Link prediction on n-ary relational facts: A graph-based approach. In *Proceedings of Findings of ACL-IJCNLP*, pages 396–407.

Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. 2016. On the representation and embedding of knowledge bases beyond binary relations. In *Proceedings of IJCAI*, pages 1300–1307.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break It Down: A Question Understanding Benchmark. In *TACL Journal*, volume 8, pages 183–198.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of ACL*, pages 643–648.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of ACL*, pages 571–581.

Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. 2021. Neural-symbolic reasoning on knowledge graphs. In *AI Open Journal*.

Liwen Zhang, John Winn, and Ryota Tomioka. 2016. Gaussian attention model and its application to knowledge base embedding and question answering. In *arXiv preprint arXiv:1611.02266*.

Richong Zhang, Junpeng Li, Jiajie Mei, and Yongyi Mao. 2018a. Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding. In *Proceedings of WWW*, pages 1185–1194.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. 2018b. Variational reasoning for question answering with knowledge graph. In *Proceedings of AAAI*, pages 6069–6076.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. In *Proceedings of ACL*, pages 2010–2022.

A Methodology Details

A.1 NL Fact Tree Construction Algorithm

The construction of NL fact tree is summarized in Algorithm 1. The input is the NL question and two empty node stacks. The output is the NL fact tree. We initialize the NL fact tree as a syntax tree, which is parsed from **Q** (Line 2). One of the empty node stack $\mathbb{V}$ stores the nodes of FT in the order of breadth first searching (Line 3). The other stack $\mathbb{V}'$ reverse this order in Line 4-10, so that the pruning (Line 12-22) is from the the bottom to the top as well as from the right to the left. The entire
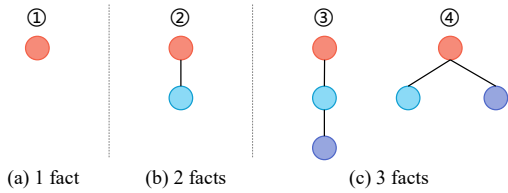
Figure 8: The fact combination modes.

elimination process does not involve the leaf nodes and their parents nodes (Line 6-10).

A.2 Score Amplification Mechanism

We devise a simple method to evaluate whether an entity is able to satisfy the upper-layer fact. For an entity e, the related predicate or attribute in the upper-layer fact is p or a. We retrieve in KG, whether there is an fact of entity e associated with p or a. If there is, then we consider that entity e is satisfying the upper-layer fact, and vice versa.

B Experimental Details

B.1 Dataset Construction

In this work, we develop an n-ary KGQA dataset: WikiPeopleQA, in which questions involve multiple n-ary facts and the background KG is also composed of n-ary facts. The specific construction process is as follows:

i) We selected WikiPeople (Guan et al., 2019) as the background KG. This n-ary KG is constructed based on Wikidata[5] and consists of character facts, e.g., *Marie Curie received Nobel Prize in Chemistry in 1911.*

ii) To build complex questions involving multiple facts, we set the maximum number of facts in a question to three, and set four fact combinations modes in advance, as shown in Figure 8. As the number of facts increases, the fact combination mode becomes more complicated.

iii) Based on the fact combination modes, we sampled a large number of fact combinations from KG. We masked off the entities in the fact combinations, and extracted the frequently occurring fact combinations. We transformed the frequent fact combinations to question templates. Here we have constructed a total of 33 question templates, listed in Table 5.

iv) We populated the entities into the question templates according to KG. Inspired by the construction process of PathQuestion (Zhou et al., 2018), in order to enrich the problematic syntactic structure and surface wording, we replaced the phrases and words in the question with synonyms.

Due to the limitation of fact diversity in WikiPeople, we only considered three facts at most. In order to contribute to the progress of n-ary KGQA research, it is necessary to increase the richness of the facts to improve the question complexity. Therefore, developing more complex n-ary KGQA datasets and evaluating FacTree on more datasets are our future research directions.

B.2 Baselines

We compare our framework with a series of baselines. The following is a detail description of the baselines:

- MemNN (Sukhbaatar et al., 2015): This model adopts an memory network to store all KG facts or related Wikipedia documents in the memory units. Three embedding matrices are employed to convert the memory information and questions into vectors for similarity calculation.

- KV-MemNN (Miller et al., 2016): This model is based on MemNN. Instead of considering the whole KG facts like MemNN, it firstly stores facts in a key-value structured memory. The key-value structure is suitable for binary facts, but not for n-ary facts.

- EmbedKGQA (Saxena et al., 2020): This model follows the basic multi-hop reasoning framework and utilizes KG embedding methods to alleviate the negative impact of KG incompleteness.

- IRN-weak (Zhou et al., 2018): This model considers the whole path from the topic entity to the answer entity. It focuses on finding a path to the answer, so IRN needs a pre-labelled path record during training process.

- MINERVA (Das et al., 2018): This model uses reinforcement learning technique to perform multi-hop reasoning on KG. Taking the input natural language question, this model averages the word embeddings as the question embedding, and then walks on KG under the supervision of the question embedding, and finally arrives at the answer entity.

798

- SRN (Qiu et al., 2020): This model uses RL method to perform multi-hop reasoning on KG. It proposes a potential-based reward shaping strategy to alleviate the delayed and sparse reward problem caused by weak supervision.

- QGG (Lan and Jiang, 2020): This model generates a modified staged query graph to deal with complex questions with both multi-hop relations and constraints.

Here we explain the source of the results in Table 2. On the WikiPeopleQA dataset, for each baseline, we run the source code of each baseline that is open source or reproduced by developers.

- MemNN: https://github.com/berlino/MemNN (reproduced)

- KV-MemNN: https://github.com/lc222/key-value-MemNN (reproduced)

- EmbedKGQA: https://github.com/malllabiisc/EmbedKGQA (open source)

- MINERVA: https://github.com/shehzaadzd/MINERVA (open source)

- SRN: https://github.com/DanSeb1295/multi-relation-QA-over-KG (reproduced)

- QGG: https://github.com/lanyunshi/Multi-hopComplexKBQA (open source)

IRN-weak needs the pre-labelled path records for training, which is not applicable to the n-ary KGQA task. So we do not evaluate IRN-weak.

For two binary KGQA datasets WC2014 and PathQuestion, the results of EmbedKGQA and QGG are obtained by our own tests. Other baseline results all cited from (Qiu et al., 2020).

B.3 Training Details

Note that in the NL fact tree, there is overlap between facts. When we construct the (syntax tree, NL fact tree) training samples, the overlap will be maintained in the lower-layer fact if it belongs to the primary triple, and in the upper-layer fact if the overlap belongs to the auxiliary description. For example, in Figure 2, the overlapping part of fact 1 and fact 2 "an NBA team" belongs to the primary triple, so it is maintained in fact 2. Conversely, "in the year" is the auxiliary description and is maintained in fact 3.

C Example of NL Fact Tree Construction

Here we display a visual example of the NL fact tree construction with the question *Who joined an NBA team in Los Angeles in the year the Warriors won the NBA championship.*

Firstly, we use the Stanford Parser to generate the syntax tree (cf. Figure 9). Then we preprocess the syntax tree (cf. Figure 10) to reduce the subsequent elimination operations according to the following rules:

- Pruning the punctuation node and its parent node, e.g., node "?" and ".".

- If all the grandchildren of a NP node are leaf nodes (more than one), we prune the parents, and combine the grandchildren to a unified leaf node, whose parent is changed to the NP node. For example, the, NBA and championship in Figure 9 are combined as the NBA championship, whose parent is "NP".

- If a node has only one child and only one grandchild, which is a leaf node, we remove the child node and let the leaf node be the only child of this node. For example, who in Figure 9 will be connected directly to its grandfather node WHNP.

Next we start eliminating nodes from the the bottom to the top as well as from the right to the left. Note that we start the elimination operation from the third-to-last layer of the tree. For each selected node, we extract a subtree that contains this node (colored in red) and its neighbor nodes (colored in blue). This subtree is fed into a classifier. The output of the classifier determines whether to eliminate this node. Figure 12 shows the specific elimination process. The previously selected node will no longer be selected, e.g., the node "SBAR" in Figure 12 (f).

After the elimination process, we delete non-leaf nodes and retain the hierarchical structure of leaf nodes. For the continuous nodes in the lower-layer, set a common placeholder node in the upper-layer. For example, the continuous nodes a team, in and Los Angeles in Figure 12 (j) will be connected to a common placeholder node (i.e., the blue node in Figure 11). The interrogative pronouns, e.g., who are also replaced directly with placeholders. Now, the NL fact tree is constructed (see Figure 11). It satisfies 1) the leaf nodes are words or phrases of the question; and 2) if the leaf nodes share the same parent, they belong to the same fact.
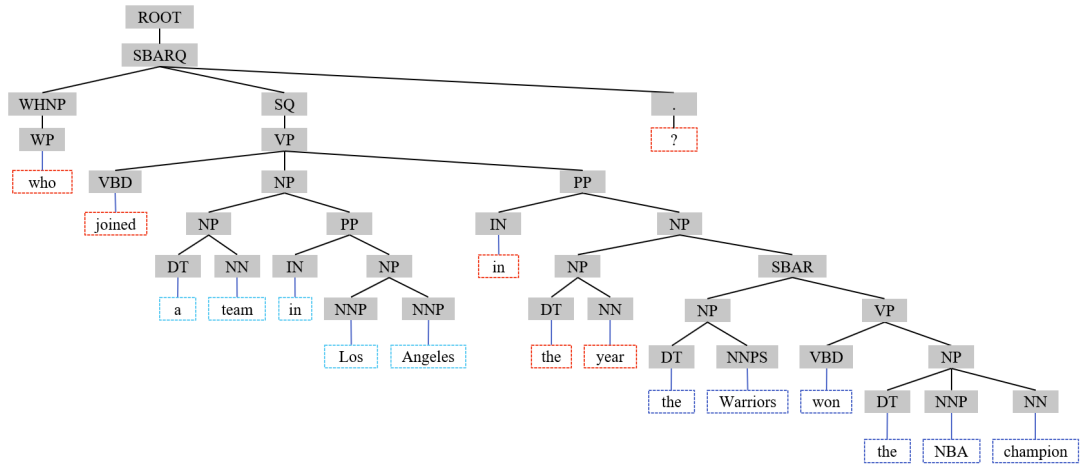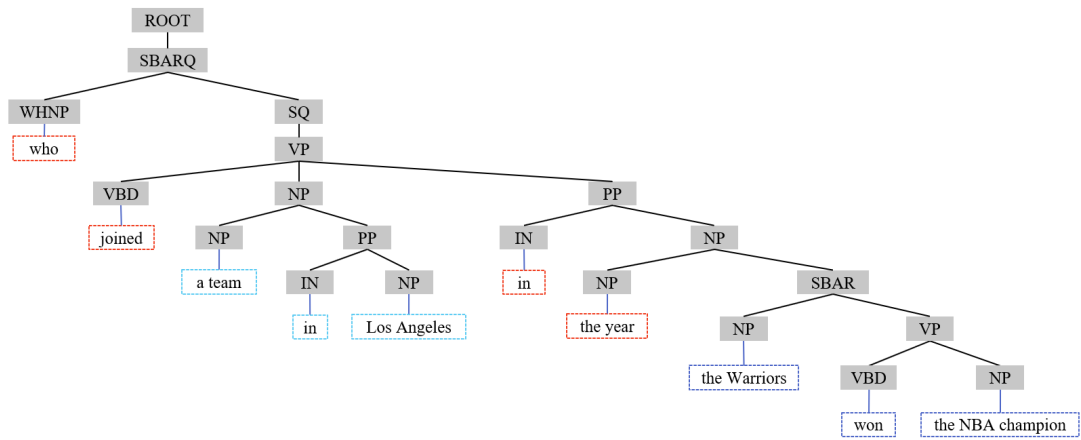
Figure 9: Syntax tree.
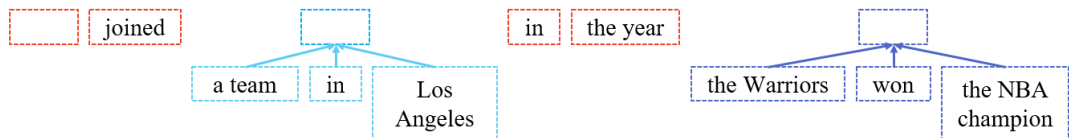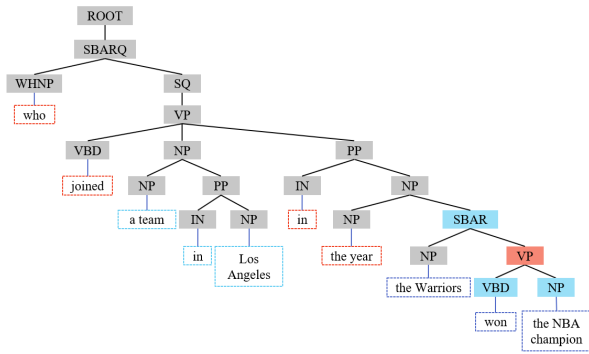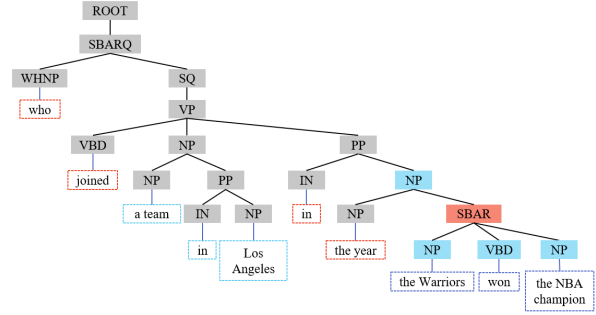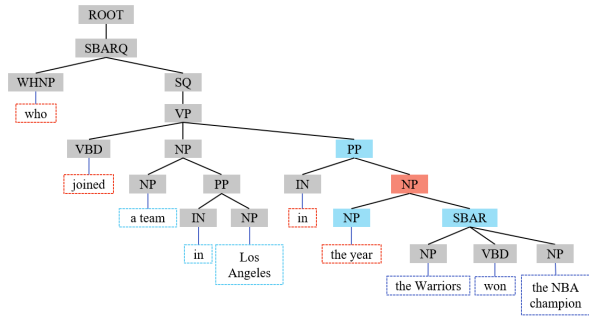

Figure 10: Syntax tree after preprocessing.

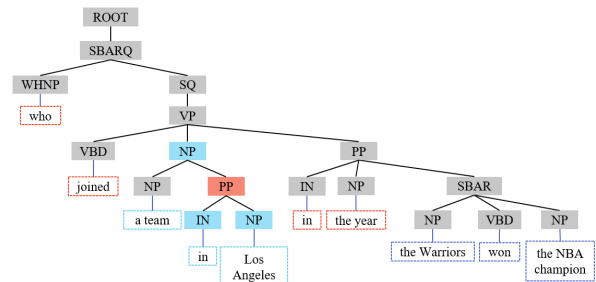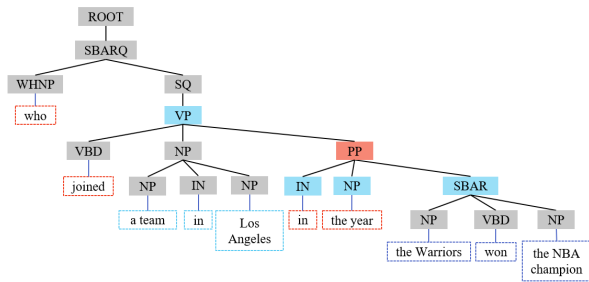Figure 11: NL fact tree. The red, blue and purple blank nodes are placeholder nodes.
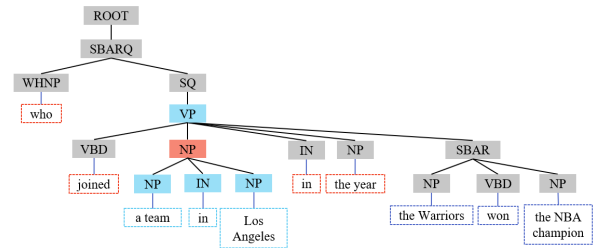
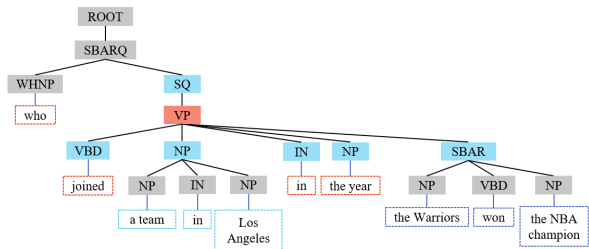(a) VP is eliminated.

(b) SBAR is retained.

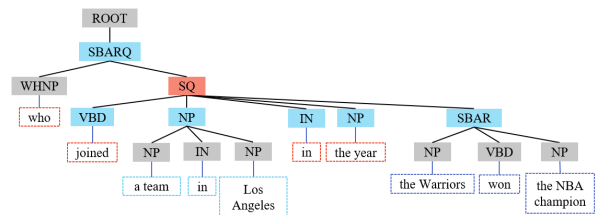(c) NP is eliminated.
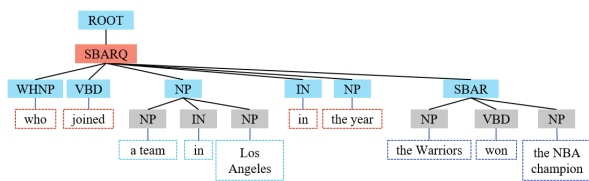
(d) PP is eliminated.

(e) PP is eliminated.
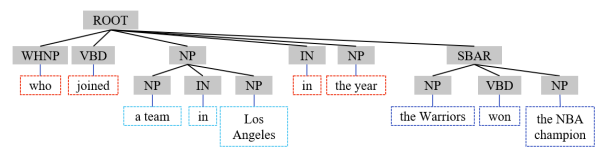
(f) NP is retained.

(g) VP is eliminated.

(h) SQ is eliminated.

(i) SBARQ is eliminated.

(j) Elimination ends.

Figure 12: Node elimination process.

ID	Template	Mode
1	who win {} award in the time {}	1
2	who is sibling of {}	1
3	what is the profession of {}	1
4	what is the country of {}	1
5	what political party did {} join	1
6	who is the spouse of {}	1
7	what is the gener of {}	1
8	who was educated at {} until {}	1
9	when did {} die	1
10	where was {} born in	1
11	who work at the place {}	1
12	who was nominated for the prize {} in the time {}	1
13	who is the mother of whose father is {}	2
14	who is the father of whose mother is {}	2
15	who win award {} in the time when {} win the {}	2
16	who is the father of who has ever won {}	2
17	who is the child of who has ever won {}	2
18	who was nominated for {} in the time when {} win the {}	2
19	what political party did the father of {} join	2
20	what is the profession of the person who has ever won the {}	2
21	who died in the place where {} born in	2
22	who born in the place where {} died in	2
23	which field did the person who has ever educated at {} work for	2
24	who is the spouse of the person who born in {}	2
25	who is the father of the person who born in {}	2
26	what is the country of the person whose father is the one has ever won the prize {}	3
27	who born in the place where the father of {} died in	3
28	who died in the place where the mother of {} born in	3
29	who was educated at the school where the person who won the prize {} was also educated at	3
30	who is the child of the person whose father is the one who is the sibling of {}	3
31	who work in the field that the person from the country {} work for	3
32	who join the political party that the person from the country {} has erver joined	3
33	when did the person from {} won the prize that {} has ever won	4
34	who joined a team in {} in the year {} won the NBA championship	4

Table 5: List of question templates and their fact combination mode. The curly braces {} indicate the entities to be filled.