Testing Logical Consistency in Question Answering Systems via Knowledge Graph-guided Metamorphic Relations

Anonymous ACL submission

Abstract

Question answering system testing methods typically generate test data by constructing metamorphic relation to evaluate system robustness. However, existing approaches mainly focus on text-level perturbations and struggle to generate test cases that require multi-step reasoning. As a result, they provide limited coverage of reasoning capabilities in QA systems. To address this limitation, this paper proposes a **K**nowledge **G**raph-based testing method for **QA** systems (KGQA). The method constructs a knowledge graph from the original test input and extracts reasoning chains as interpretable evidence. Based on these chains, two types of semantically equivalent metamorphic relations, namely context insertion and adjacent fact substitution, are designed to generate derived questions with multi-hop reasoning variations. Experimental results show that, on the HotpotQA and NarrativeQA datasets, the violation rate of the test cases generated by KGQA improves by an average of 5.98% and 4.5%, respectively, compared with baseline methods.

1 Introduction

With the rapid advancement of Natural Language Processing (NLP), Question Answering (QA) systems have been widely deployed in intelligent applications such as [1], Cortana[2], and Alexa[3]. However, similar to other NLP systems, including machine translation (Zhong et al., 2021; Sun et al., 2020), QA systems can still generate incorrect answers in real-world scenarios. Such errors may result in economic losses (Këpuska and Bohouta, 2018) and can lead to severe consequences in high-risk domains, such as healthcare and law (Schramowski et al., 2022). For instance, Babylon Health once provided incorrect answers that led to misdiagnosis, thereby delaying appropriate treatment for patients. Therefore, rigorous testing of QA systems is essential.

Existing testing methods can be broadly categorized into reference-based approaches and metamorphic relation (MR)-based approaches (Shen et al., 2026). Reference-based methods rely on manually annotated data and evaluate system performance by comparing model outputs with ground-truth answers (Longpre et al., 2021; Jia and Liang, 2017; Ribeiro et al., 2020). However, these methods face several limitations, including high annotation costs, limited data scale, and insufficient coverage of real-world scenarios (Chen et al., 2021).

To reduce reliance on annotated data, approaches such as QAAskeR (Chen et al., 2021) and QATest (Liu et al., 2022) adopt metamorphic relations to construct semantically equivalent questions. These methods verify whether the outputs before and after transformation satisfy predefined relations, thus enabling defect detection without requiring manually labeled answers. For example, QAAskeR transforms question–answer pairs into declarative sentences and generates related questions, while QATest applies data augmentation techniques to produce semantically equivalent variants of the original questions. However, these methods mainly focus on evaluating semantic understanding and offer limited coverage of complex logical reasoning scenarios. As a result, they are less effective in identifying defects related to reasoning consistency. In addition, their transformation strategies often rely on syntactic perturbations or noise injection, which may distort the original semantics and reduce the quality of the generated test cases.

To address the above limitations, we propose a **K**nowledge **G**raph-based testing method for **QA** systems (KGQA). The proposed method first extracts entities and relations from the original question and constructs a knowledge graph through iterative clustering. It then derives reasoning chains of

[1] https://www.apple.com/siri
[2] https://www.microsoft.com/en-us/cortana
[3] https://alexa.amazon.com

predefined length as interpretable evidence. Based on these reasoning chains, two types of metamorphic relations are designed: context insertion and adjacent fact substitution. These transformations generate semantically equivalent test inputs, enabling KGQA to evaluate the logical consistency of QA systems. Experimental results show that KGQA generates more error-triggering test cases on the HotpotQA (Yang et al., 2018) and NarrativeQA (Kočiský et al., 2018) datasets, outperforming all baseline methods. Compared with QAAskeR+, which achieves the best performance on HotpotQA, and QAQA, which performs best on NarrativeQA, KGQA improves the violation rate by 5.98% and 4.5%, respectively.

2 The Method

We present a **K**nowledge **G**raph-based testing method for **QA** systems (KGQA). The overall framework is illustrated in Figure 1 and consists of four stages: knowledge graph construction, reasoning chain extraction, metamorphic relation design, and answer verification. First, KGQA employs a large language model to extract entities and relations from the original test inputs and constructs a knowledge graph through iterative clustering. Next, a backtracking depth-first search (B-DFS) algorithm is used to extract reasoning chains with controllable depth. Then, Boolean questions are constructed based on the triples in the reasoning chains, and derived questions are generated using semantically equivalent metamorphic relations. Finally, logical consistency is evaluated by comparing the answers to the original and derived questions, thereby identifying potential defects in the system.

2.1 Knowledge Graph Construction

The test input of a QA system is represented as $t = \{q, c\}$ (Shen et al., 2026), where q denotes the question and c denotes the context that provides the information required to answer q. KGQA focuses on constructing test inputs with explicit reasoning relationships. Such relationships rely on factual associations within the context. However, directly using unstructured text to model multi-hop relations may introduce semantic noise and makes it difficult to control the reasoning process. Therefore, a structured representation is needed to support interpretable reasoning.

To this end, we introduce a knowledge graph $G = (E, R, F)$, where E is the set of entities, R is the set of relations, and F is the set of factual triples. Each factual triple $f \in F$ is defined as follows:

$$f = (h, t, r), h, t \in E, r \in R. \quad (1)$$

Here, h and t denote the head and tail entities, respectively, and r represents the relation between them. By explicitly modeling entity relationships, the knowledge graph provides a clear representation of multi-hop reasoning paths and reduces semantic noise. Based on this representation, KGQA transforms the context of the original test input into a knowledge graph, which serves as the foundation for subsequent reasoning chain extraction.

In terms of construction, traditional methods (e.g., OpenIE (Angeli et al., 2015) and GraphRAG[4] often produce duplicated entities and ambiguous relations, leading to redundant and sparse graph structures (Mo et al., 2025). To address this issue, KGQA adopts the KGGen method based on large language models (Mo et al., 2025). This method employs a two-stage extraction process and an iterative clustering strategy. It effectively reduces redundancy and converts the original context into a compact and well-structured knowledge graph G.

2.2 Reasoning Chain Extraction

To construct test inputs for evaluating the logical reasoning ability of QA systems, KGQA performs controlled-depth logical expansion based on the original test inputs. To avoid semantic noise and circular reasoning caused by unconstrained expansion, we extract reasoning chains of bounded length and without cycles from the knowledge graph $G = (E, R, F)$ constructed in Section 2.1 (Li et al., 2024). A reasoning chain is defined as a multi-hop path $p(e_s, e_t)$ from a head entity e_s to a tail entity e_t in the knowledge graph. Its quality is constrained by two key factors. Directed acyclicity ensures that a reasoning chain does not contain repeated entities, as such repetitions may introduce circular reasoning and redundant paths, thereby violating the unidirectional nature of reasoning. In addition, a semantic boundary constraint arises as the hop count increases , since the semantic relevance between entities gradually decreases (Peng et al., 2021).

Based on these constraints, reasoning chain extraction is formulated as a hop-constrained simple

[4] Microsoft Research blog

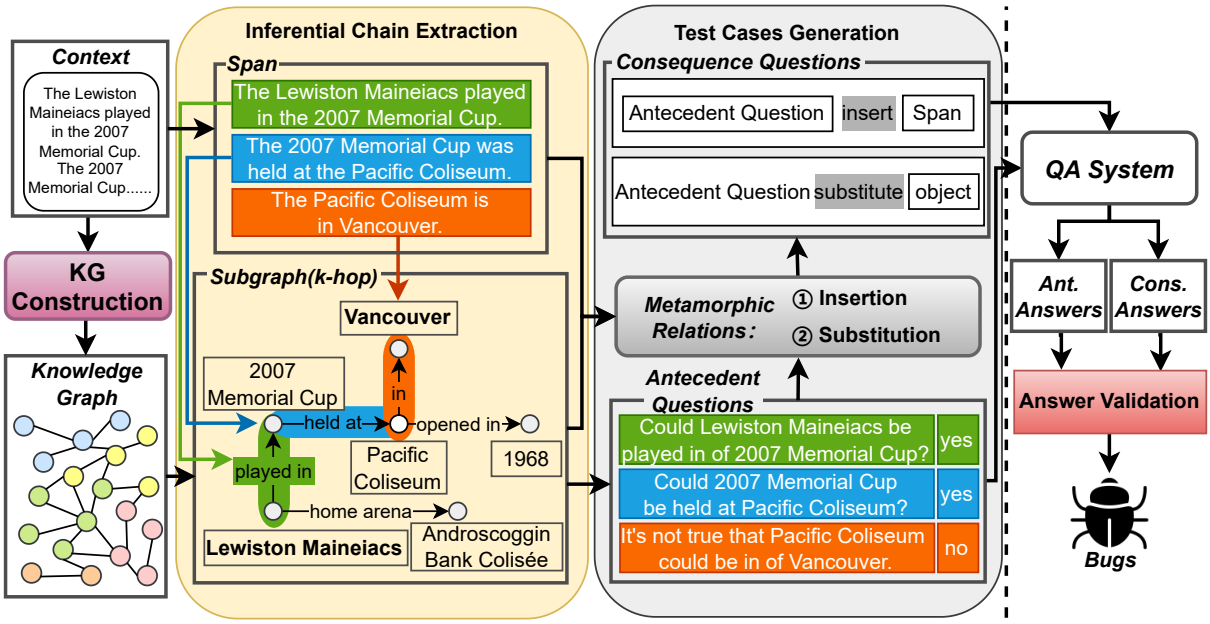

Figure 1: Overall framework of KGQA.

path enumeration problem, in which all vertices in a path of length k are distinct. Although this problem has high computational complexity in large graphs (Bui, 2023), the knowledge graph used in this work is derived from a single document and is relatively small. Therefore, enumeration can be performed with acceptable computational cost.

To solve this problem, we adopt a backtracking-based depth-first search strategy and implement a Backtracking DFS (B-DFS) algorithm. The algorithm limits the search depth using a predefined hop number k and updates node visit states during the search to prune repeated vertices, thereby ensuring the simple-path property.

The procedure is as follows. During the search, B-DFS maintains a stack S_{vis} to record the nodes visited in the current path, with the initial state $S_{\text{vis}} = \langle s \rangle$. Given the current node s and a candidate neighbor $e \in E$, pruning strategies are applied before expanding the path. Depth pruning ensures that when $|S| - 1 = k$, the depth limit is reached and backtracking is triggered, while the search proceeds only when $|S| - 1 < k$. Cycle pruning prevents revisiting nodes already in the current path; if $e \in S_{\text{vis}}$, further expansion would create a cycle, and the branch is pruned immediately.

When the depth termination condition is met, B-DFS maps the node sequence in the stack to a reasoning chain and records the corresponding sequence of relations, producing a structured output. The algorithm then performs a pop operation to backtrack and continues exploring other branches until all paths have been traversed. Finally, a set of reasoning chains $P = \{p_1, p_2, \ldots, p_m\}$ is obtained.

After the reasoning chains are extracted, KGQA converts single-hop triples into Boolean original questions based on predefined templates. For example, given a triple (e_0, e_1, r_1), a question such as "Is it true that e_0 r_1 e_1?" can be constructed. These original questions serve as the initial test inputs and provide the basis for the subsequent generation of derived questions. The proposed algorithm extracts potential multi-hop reasoning relations between entities under a given hop constraint. This process provides the data foundation for constructing metamorphic relations in the next stage.

2.3 Metamorphic Relation Construction

To enable effective testing of QA systems without manually annotated reference answers, we adopt metamorphic testing. It constructs metamorphic relations that satisfy domain knowledge constraints, transforms original test inputs into derived inputs, and checks whether the outputs before and after transformation satisfy the expected relations. This approach reduces the need for manually constructed test cases. Based on the reasoning chains extracted in Section 2.2, KGQA designs two types of semantically equivalent metamorphic relations: context insertion (MR1) and adjacent fact substitution (MR2).

MR1: Context Insertion

MR1 evaluates whether a QA system can maintain stable answers when additional logically related information is introduced into the question. Monotonic reasoning refers to a reasoning process in which newly introduced facts may derive additional conclusions without invalidating previously established conclusions (Xia et al., 2001). In the reasoning chains constructed in this paper, the reasoning process follows a predefined path. The introduction of subsequent triples does not negate or modify earlier ones; instead, it extends the existing evidence. Therefore, the process remains monotonic. Under this relation, when context fragments corresponding to other triples in the reasoning chain are inserted into the original question as additional information, the QA system is expected to produce the same answer.

Specifically, given a reasoning chain $p = \langle (e_0, e_1, r_1), \ldots, (e_{k-1}, e_k, r_k) \rangle$, if the original question q_{orig} is derived from the first triple (e_0, e_1, r_1), then the context fragments corresponding to the remaining triples (e_{i-1}, e_i, r_i) are denoted as span_i. These fragments form a set $S_{\text{ins}} = \{\text{span}_i | (e_{i-1}, e_i, r_i) \in p, i = 2, \ldots, k\}$. The derived question q_{ins} is constructed by inserting S_{ins} into the original question:

$$q_{\text{ins}} = \text{Insert}(q_{\text{orig}}, S_{\text{ins}}) \tag{2}$$

Since both S_{ins} and q_{orig} are derived from facts along the same reasoning chain, inserting S_{ins} does not change the original semantics. Therefore, the QA system is expected to produce the same answer, i.e., $P(q_{\text{ins}}) = P(q_{\text{orig}})$. If this condition is not satisfied, the system is considered to violate MR1. Figure 2(a) illustrates an example of MR1. To obtain the context fragment set S_{ins}, KGQA first converts each triple (e_{i-1}, e_i, r_i) into a simple sentence of the form "e_{i-1} r_i e_i". It then retrieves the most semantically similar text fragment from the context. Since the context often contains pronouns (e.g., he, she, it), direct matching may cause ambiguity. To address this issue, KGQA applies a coreference resolution model, NeuralCoref[5], to preprocess the context by replacing pronouns with their corresponding entity mentions. Next, both the simple sentence and each sentence in the context are encoded into semantic vectors using Sentence-BERT (SBERT) (Reimers and Gurevych, 2019). Cosine similarity is then computed, and the sentence with the highest similarity is selected as the context fragment span_i corresponding to the triple.

MR2: Adjacent Fact Substitution

Errors in multi-hop reasoning often arise in tasks that require step-by-step inference over multiple facts (Cheng et al., 2025). While models generally perform well on single-hop reasoning, they tend to exhibit reasoning inconsistencies when multiple related facts are combined. MR2 addresses this issue by exploiting shared entities between adjacent triples in a reasoning chain. It replaces the shared entity with a logically equivalent reference, thereby constructing semantically consistent reasoning questions. Specifically, given two adjacent triples $(e_{i-1}, e_i, r_i)$and (e_i, e_{i+1}, r_{i+1}), they share the entity e_i and form a reasoning relation $\langle e_{i-1} \xrightarrow{r_i} e_i \xrightarrow{r_{i+1}} e_{i+1} \rangle$. The representation of the shared entity e_i can be replaced with an entity description that has relation r_{i+1} with e_{i+1}. Since this replacement is logically equivalent under the relation constraint, it preserves the semantics of the original question. Therefore, the QA system is expected to produce the same answer.

The substitution process proceeds as follows. Given adjacent triples (e_0, e_1, r_1) and (e_1, e_2, r_2), the original question is "Is it true that e_0 r_1 e_1?". KGQA constructs a restrictive relative clause $RRC(e_1)$ based on the reasoning relation $\langle e_1 \xrightarrow{r_2} e_2 \rangle$. The construction is defined as:

$$RRC(e_i) = \text{the entity that } r_2\ e_2 \tag{3}$$

The entity e_1 in the original question is then replaced by $RRC(e_i)$ to form a derived question q_{sub}:

$$q_{\text{sub}} = \text{Substitute}(q_{\text{orig}}, RRC(e_i)) \tag{4}$$

The resulting question becomes "Is it true that e_0 r_1 $RRC(e_i)$?". Because the substitution is logically equivalent, the QA system should produce the same answer for both questions, i.e., $P(q_{\text{ins}}) = P(q_{\text{orig}})$. Figure 2(b) presents an example of MR2. Note that relations in a knowledge graph are semantic labels and may not follow natural language grammar. Directly using them as predicates can lead to ungrammatical expressions. To address this issue, this work employs a grammar correction model[6] based on FLAN-T5-large (Chung et al., 2024) to refine the generated

[5] https://github.com/huggingface/neuralcoref

[6] https://huggingface.co/pszemraj/flan-t5-large-grammar-synthesis

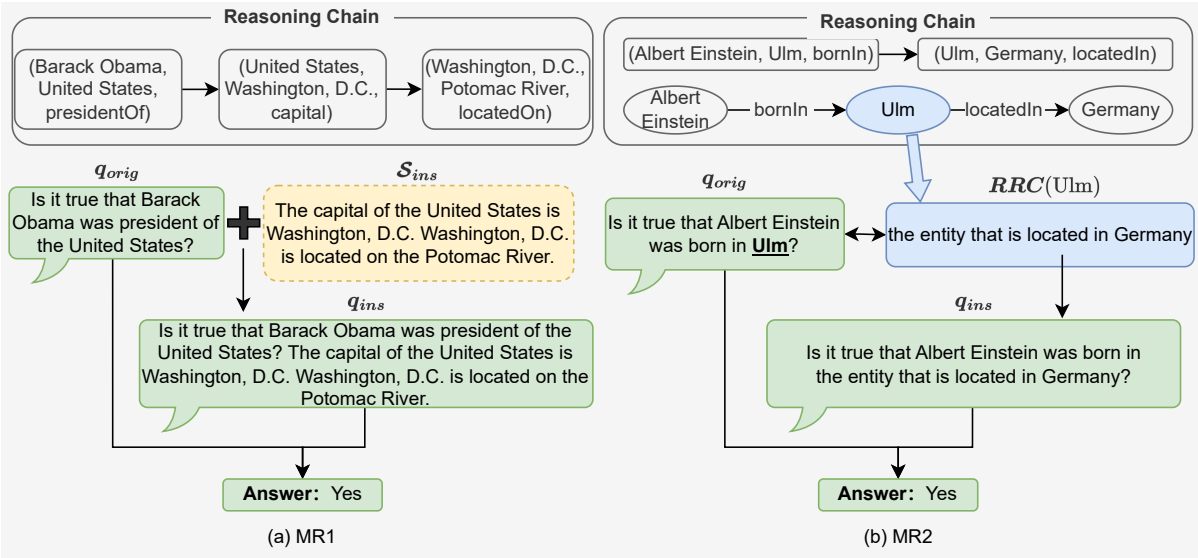

Figure 2: Examples of Metamorphic Relations.

questions. The model maps each relation r into a natural language predicate, ensuring that the generated questions are grammatically correct.

2.4 Answer Verification

After obtaining the original questions and their corresponding derived questions, KGQA inputs each question pair into the target QA system and verifies whether the outputs remain consistent. Since the generated questions are Boolean (yes/no), this study introduces answer normalization and semantic matching strategy to reduce the impact of output variations.

Specifically, the model outputs are first preprocessed by removing non-alphabetic characters and converting all text to lowercase. If an answer contains explicit affirmative or negative keywords (e.g., "yes" or "no"), it is directly mapped to the corresponding label. For free-text answers that do not contain such keywords, the Sentence-BERT (SBERT) model (Reimers and Gurevych, 2019) is used to compute the semantic similarity between the answer and the standard labels "yes" and "no". The answer is then assigned to the label with the higher similarity score. If the similarity to both labels is below a predefined threshold, the answer is considered semantically ambiguous and is excluded from subsequent analysis.

3 Experimental Design

This section describes the experimental setup, including the research questions, data preparation, target systems, and experimental settings.

3.1 Research Question

To evaluate the effectiveness of the proposed knowledge graph-based metamorphic testing method, KGQA, the following research questions are defined:

RQ1: How effective is KGQA in revealing potential defects in QA systems compared with existing methods?

RQ2: How does the quality of test cases generated by KGQA compare with those generated by existing methods?

RQ3: How does test effectiveness vary with different reasoning chain hop lengths?

3.2 Target Systems

UnifiedQA (Khashabi et al., 2020) is a QA system based on the T5 model. In this study, its latest version, UnifiedQA V2 (Khashabi et al., 2022), is used. It is pretrained on 20 datasets, including SQuAD 1.1 (Rajpurkar et al., 2016), SQuAD 2.0 (Rajpurkar et al., 2018), BoolQ (Clark et al., 2019), and MultiRC (Khashabi et al., 2018).

UnifiedQA provides model configurations with different parameter sizes. To analyze the impact of model scale on semantic understanding and complex reasoning ability, three configurations are used: UnifiedQA-v2-small[7], UnifiedQA-v2-base[8], and UnifiedQA-v2-large[9], with 60M, 220M, and

[7] https://huggingface.co/allenai/unifiedqa-t5-small
[8] https://huggingface.co/allenai/unifiedqa-t5-base
[9] https://huggingface.co/allenai/

770M parameters, respectively.

3.3 Baseline Methods

To evaluate the effectiveness of KGQA, it is compared with three QA testing methods: QATest (Liu et al., 2022), QAQA (Shen et al., 2022), and QAAskeR+ (Xie et al., 2023).

QATest generates multiple input variants by introducing different types of input transformations, such as synonym replacement and text reordering, to evaluate the robustness of QA systems under unknown or abnormal inputs.

QAQA adopts a semantic-guided search strategy. It retrieves sentences that are semantically similar to the original input from the training data as candidate insertion text, and performs sentence-level mutations to construct new test cases.

QAAskeR+ extends QAAskeR by adding two knowledge-enhanced metamorphic relations, which generate special interrogative sentences and yes/no questions on top of the original three relations.

3.4 Datasets

The experiments use HotpotQA (Yang et al., 2018) and NarrativeQA (Kočiský et al., 2018) as datasets.

HotpotQA is a large-scale multi-hop QA dataset based on Wikipedia. It is used to evaluate a model's ability to perform complex reasoning across multiple documents and to provide explanatory predictions.

NarrativeQA is an abstractive QA dataset, where answers are provided in free-text form and are not restricted to contiguous spans from the reference context.

3.5 Evaluation Metrics

The experiments employ three evaluation metrics: number of violations, violation rate, and precision. These metrics are commonly used to assess the effectiveness of test cases in QA system testing (Shen et al., 2022; Ji et al., 2025).

The number of violations refers to the number of test cases that trigger incorrect answers from the QA system. A test case is considered a violation if it does not satisfy the predefined metamorphic relations.

The violation rate is defined as the ratio of violating test cases to the total number of generated test cases. It measures how effectively a testing method exposes defects.

Precision is defined as the proportion of violating test cases that correspond to actual defects. In the experiment, 200 samples are randomly selected from the violating test cases identified by each method. These samples are manually annotated to determine whether the QA system outputs represent true defects. Precision is then calculated as follows:

$$\text{Precision} = \frac{\text{Number of real defects}}{\text{Total number of sampled violating test cases}} \quad (5)$$

4 Experimental Results and Analysis

4.1 Results for RQ1

To evaluate the effectiveness of KGQA in triggering errors in QA systems, we compare it with three baseline methods: QATest, QAQA, and QAAskeR+. Unlike the baseline methods, KGQA does not directly use questions from QA datasets as the original test inputs. Instead, it generates questions based on reasoning chains.

To ensure a fair comparison, a unified setting is adopted. Specifically, the original questions extracted by KGQA are used as the initial seeds for QATest, QAQA, and QAAskeR+. This design eliminates the impact of input differences on the experimental results. Table 1 reports the number of test cases generated by each method that violate metamorphic relations. The values in parentheses indicate the corresponding violation rate.

On the HotpotQA dataset, compared with QATest, QAQA, and QAAskeR+, KGQA detects, on average, 10,065, 10,039, and 11,152 more test cases that violate metamorphic relations across the three models. At the same time, its violation rate increases by 11.9%, 3.4%, and 2.63%, respectively. These results indicate that the complex reasoning questions constructed by KGQA can effectively expose more defects in QA systems. On the NarrativeQA dataset, KGQA improves the violation rate by 6.56%, 2.83%, and 3.93% compared with QATest, QAQA, and QAAskeR+, respectively. Although the improvements over QAQA and QAAskeR+ are relatively modest, the number of violating test cases generated by KGQA is 4.5 times and 8.8 times higher than those methods, respectively.

In addition, based on the precision metric defined in Section 3.5, KGQA achieves an average precision of 98.5%, which is substantially higher than QATest (66%), QAQA (93.5%), and QAAskeR+

unifiedqa-t5-large

Dataset	Method	UnifiedQA-v2 small	UnifiedQA-v2 base	UnifiedQA-v2 large
HotpotQA	QATest	2,950(27.3%)	2,579(22.8%)	2,137(18.9%)
	QAQA	2,900(34.7%)	2,535(31.3%)	2,308(28.5%)
	QAAskeR+	1,640(36.1%)	1,467(32.2%)	1,298(28.5%)
	KGQA	**14,500(39.0%)**	**12,403(34.9%)**	**10,959(30.8%)**
NarrativeQA	QATest	1,461(33.8%)	1,241(28.7%)	1,076(24.9%)
	QAQA	1,575(37.3%)	1,364(32.3%)	1,225(29.0%)
	QAAskeR+	775(35.5%)	692(31.7%)	613(28.1%)
	KGQA	**6,919(40.3%)**	**6,164(35.9%)**	**5,306(30.9%)**

Table 1: Number of test cases violating metamorphic relations generated by each method.

(67.5%). This result is attributed to the fact that the transformations used in KGQA preserve the semantics of the original questions. As a result, the generated test inputs more reliably satisfy the predefined metamorphic relations, which improves the reliability of the testing outcomes.

4.2 Results for RQ2

To evaluate whether KGQA reduces the generation of invalid test cases, the quality of the generated test cases is assessed from two aspects: validity and grammatical correctness. Following prior work (Ji et al., 2025), validity is defined as whether a generated test case satisfies the predefined metamorphic relations. In this section, for each of the four methods, 200 test cases that violate metamorphic relations are randomly sampled from each dataset, resulting in a total of 800 test cases (200 × 4). Two authors independently evaluate each test case in terms of validity and grammatical correctness. A test case is labeled as 1 if it satisfies the corresponding criterion and 0 otherwise. The types of errors are also recorded. The results of the manual evaluation are presented in Table 2.

From Table 2, it can be observed that the test cases generated by KGQA exhibit high overall quality and perform well in both validity and grammatical correctness. First, in terms of validity, most test cases generated by KGQA satisfy the predefined metamorphic relations. The validity of MR1 and MR2 reaches 100% and 97%, respectively, both higher than those of existing methods. This result indicates that the metamorphic relations designed in KGQA preserve semantic equivalence under the logical constraints of reasoning chains, thereby reducing invalid test cases caused by semantic drift. Second, in terms of grammatical correctness, KGQA also achieves strong performance. The grammatical correctness of test cases generated by MR1 and MR2 is 91% and 83%, respectively. This demonstrates that the reasoning chain-based transformations not only ensure semantic validity but also maintain proper linguistic form.

4.3 Results for RQ3

The objective of RQ3 is to analyze the impact of reasoning chain length on the effectiveness of test cases in revealing defects in QA systems. Under the same experimental settings, we evaluate the effect of different hop numbers k by varying only the length of the reasoning chains. It should be noted that when k=1, the reasoning chain contains only a single triple and does not involve a reasoning process. Therefore, the experiments start from k=2. Table 3 reports the number of generated test cases, the number of detected violations, and the violation rate of KGQA on the HotpotQA dataset under different values of k.

As shown in Table 3, both the total number of generated test cases and the number of violations decrease as the hop count k increases. This trend occurs because, as k grows, the number of reasoning chains that satisfy the constraints is significantly reduced. Consequently, fewer test cases are generated, and fewer errors are triggered in the QA system. In contrast, the violation rate increases with larger k. Reasoning chains with higher hop numbers typically involve more entities and relations, resulting in more complex structures. This increased complexity makes it more challenging for QA systems to perform correct reasoning, thereby increasing the likelihood of errors. To better illustrate the trade-off between these two metrics, Min–Max normalization is applied to both indicators, and their trends with respect to k are plotted. Figure 3 presents the normalized curves of the num-

Method	QATest	QAQA	QAAskeR+	KGQA	
				MR1	MR2
Validity	66%	93.5%	67.5%	100%	97%
Grammatical correctness	97%	88%	85.5%	91%	83%
Cohen's Kappa	83.7%	86.1%	80.3%	92.1%	78.4%

Table 2: Results of manual evaluation.

Hop number (k)	Total number of test cases	Number of violations	Violation rate
2	412,218	122903	29.8%
3	194,664	60080	30.8%
4	85,600	27726	32.3%
5	35,582	12403	34.8%
6	14,394	5254	36.5%
7	6,434	2507	38.9%
8	2,582	1029	39.8%
9	948	401	42.3%
10	320	147	45.9%

Table 3: Test cases generated with different hop count k.

Figure 3: Trends of the number of violations and the violation rate under different hop numbers k.

ber of violations and the violation rate.

As shown in Figure 3, the two curves intersect between k=4 and k=5, indicating a relative balance between the two metrics in this range. To balance testing scale and defect detection efficiency, this study selects k=5 as the final reasoning chain length.

5 Conclusion

To address the limitations of existing methods in detecting logical reasoning errors and their tendency to generate invalid test cases, we propose a knowledge graph-based testing method, KGQA. The method constructs a knowledge graph and extracts reasoning chains with controllable length. It then designs two types of metamorphic relations, namely context insertion and adjacent fact substitution, to generate multi-hop reasoning test cases while preserving semantic equivalence. These test cases are used to evaluate the logical consistency of QA systems. Experimental results demonstrate that KGQA outperforms existing methods across different sizes of UnifiedQA models. It improves the violation rate by 8% and 8.5% on the HotpotQA and NarrativeQA datasets, respectively.

Limitations

Although KGQA demonstrates effectiveness in testing the logical consistency of QA systems, this work still has two limitations.

First, the quality of the generated test cases depends on the quality of the constructed knowledge graph. If the knowledge graph fails to capture deeper logical relations, the number of valid reasoning chains decreases rapidly as the hop number increases, making extremely long or highly complex reasoning processes difficult to capture effectively.

Second, to ensure reliable answer verification, this work only considers boolean-style questions with yes/no answers. Although this setting simplifies consistency checking, it limits the diversity of generated test cases. Future work could explore how to generate questions with more diverse answer types while maintaining the reliability of answer verification.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Thai Bui. 2023. Finding a path of length k: An expository. *Computing Research Repository*, arXiv:2304.08300. Version 1.

Songqiang Chen, Shuo Jin, and Xiaoyuan Xie. 2021. Testing your question answering software via asking recursively. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 104–116.

Lingyun Cheng, Yinzhang Guo, and Qingfang Liu. 2025. Multi-hop knowledge reasoning based on adversarial reinforcement learning. *Pattern Recognition and Artificial Intelligence*, 38(1):22–35.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Pin Ji, Yang Feng, Ruohao Zhang, Ruichen Xue, Yichi Zhang, Weitao Huang, Jia Liu, and Zhihong Zhao. 2025. NLPLego: Assembling test generation for natural language processing applications. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–36.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Veton Këpuska and Gamal Bohouta. 2018. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 99–103.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Daniel Khashabi, Yeganeh Kordi, and Hannaneh Hajishirzi. 2022. UnifiedQA-v2: Stronger generalization via broader cross-format training. *Computing Research Repository*, arXiv:2202.12359. Version 1.

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

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Bangzheng Li, Ben Zhou, Fei Wang, Xingyu Fu, Dan Roth, and Muhao Chen. 2024. Deceptive semantic shortcuts on reasoning chains: How far can models go without hallucination? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7675–7688, Mexico City, Mexico. Association for Computational Linguistics.

Zixi Liu, Yang Feng, Yining Yin, Jingyu Sun, Zhenyu Chen, and Baowen Xu. 2022. QATest: A uniform fuzzing framework for question answering systems. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7052–7063, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Belinda Mo, Kyssen Yu, Joshua Kazdan, Proud Mpala, Lisa Yu, Charilaos Kanatsoulis, and Sanmi Koyejo. 2025. KGGen: Extracting knowledge graphs from plain text with language models. In *Advances in Neural Information Processing Systems*, pages 30092–30115.

You Peng, Xuemin Lin, Ying Zhang, Wenjie Zhang, Lu Qin, and Jingren Zhou. 2021. Efficient hop-constrained s-t simple path enumeration. *The VLDB Journal*, 30(5):799–823.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Patrick Schramowski, Cigdem Turan, Nico Andersen, Constantin A. Rothkopf, and Kristian Kersting. 2022. Large pre-trained language models contain human-like biases of what is right and wrong to do. *Nature Machine Intelligence*, 4(3):258–268.

Qingchao Shen, Junjie Chen, Jie M. Zhang, Haoyu Wang, Shuang Liu, and Menghan Tian. 2022. Natural test generation for precise testing of question answering software. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12.

Qingchao Shen, Xingjian Li, Jiajun Jiang, Junjie Chen, Yixian Qi, and Zan Wang. 2026. Logical reasoning testing of intelligent question answering system. *Journal of Software*, 37(2):543–562.

Zeyu Sun, Jie M. Zhang, Mark Harman, Mike Papadakis, and Lu Zhang. 2020. Automatic testing and improvement of machine translation. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 974–985.

Youming Xia, Youan Xia, Tianwei Xu, and Jingxiu Zhao. 2001. Research on domain expert knowledge bases based on non-monotonic reasoning. *Computer Science*, (9):106–108.

Xiaoyuan Xie, Shuo Jin, and Songqiang Chen. 2023. qaAskeR+: a novel testing method for question answering software via asking recursive questions. *Automated Software Engineering*, 30(1):14.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Wenkang Zhong, Jidong Ge, Xiang Chen, Chuanyi Li, Ze Tang, and Bin Luo. 2021. Multi-granularity metamorphic testing for neural machine translation system. *Journal of Software*, 32(04):1051–1066.