Mitigating Large Language Model Hallucinations via Autonomous Knowledge Graph-Based Retrofitting

Xinyan Guan\textsuperscript{1,2,}\textsuperscript{*}, Yanjiang Liu\textsuperscript{1,2,}\textsuperscript{*}, Hongyu Lin\textsuperscript{1}, Yaojie Lu\textsuperscript{1,\dagger}, Ben He\textsuperscript{1,2}, Xianpei Han\textsuperscript{1,3}, Le Sun\textsuperscript{1,2,\dagger}

\textsuperscript{1}Chinese Information Processing Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China
\textsuperscript{2}University of Chinese Academy of Sciences, Beijing, China
\textsuperscript{3}State Key Laboratory of Computer Science Institute of Software, Chinese Academy of Sciences, Beijing, China

\{guanxinyan2022, hongyu, luyaojie, xianpei, sunle\}@iscas.ac.cn,
luyanjiang22@mails.ucas.ac.cn, benhe@ucas.edu.cn

Abstract

Incorporating factual knowledge in knowledge graph is regarded as a promising approach for mitigating the hallucination of large language models (LLMs). Existing methods usually only use the user’s input to query the knowledge graph, thus failing to address the factual hallucination generated by LLMs during its reasoning process. To address this problem, this paper proposes Knowledge Graph-based Retrofitting (KGR), a new framework that incorporates LLMs with KGs to mitigate factual hallucination during the reasoning process by retrofitting the initial draft responses of LLMs based on the factual knowledge stored in KGs. Specifically, KGR leverages LLMs to extract, select, validate, and retrofit factual statements within the model-generated responses, which enables an autonomous knowledge verifying and refining procedure without any additional manual efforts. Experiments show that KGR can significantly improve the performance of LLMs on factual QA benchmarks especially when involving complex reasoning processes, which demonstrates the necessity and effectiveness of KGR in mitigating hallucination and enhancing the reliability of LLMs.

Introduction

Large Language Models (LLMs) have gained increasing prominence in artificial intelligence. The emergence of potent models such as ChatGPT (OpenAI 2022) and LLaMA (Touvron et al. 2023) has led to substantial influences on many areas like society, commerce, and research. However, LLMs still suffer from severe \textit{factual hallucination} problems, i.e., LLMs can frequently generate unsupported false statements regarding factual information due to their lack of intrinsic knowledge (Ji et al. 2023). For example, in Figure 1, ChatGPT fails to provide an accurate response to the query “\textit{When is Frédéric Chopin’s father’s birthday?}” due to a wrong belief that Nicolas Chopin’s birthday is on June 17, 1771. Factual hallucination poses a severe challenge for LLM applications, particularly in real-world situations where factual accuracy holds significance. Consequently, the endeavor to alleviate factual hallucinations in LLMs has become a research hotspot in NLP field (Liu et al. 2021; Kang and Hashimoto 2020).

On the other hand, Knowledge Graphs (KGs) store a substantial amount of high-quality factual information, which can significantly alleviate factual hallucination if incorporated with LLMs. For example, in Figure 1, we can retrofit the erroneous statement “\textit{Nicolas Chopin was born on June 17, 1771}” by referring to the provided factual knowledge “(Nicolas Chopin, date of birth, 1771-04-15T00:00:00)" in Wikidata. Recent work has focused on integrating LLMs with KGs by retrieving the entities in the query within knowledge graphs. Then the obtained factual triples are utilized as an additional context for LLMs to enhance their factual knowledge (Baek, Aji, and Saffari 2023; Chase 2022). Unfortunately, these approaches are limited to retrieving factual knowledge relevant to entities explicitly mentioned within the given query. However, the fundamental capability of large language models involves intricate and multi-step reasoning. Such reasoning processes often necessitate the validation and augmentation of factual knowledge that may be employed during the reasoning process. For example, in the case shown in Figure 1, LLM fails to answer the question because it requires an intermediate knowledge about “\textit{Nicolas Chopin was born on April 15, 1771}”. However, such information does not refer to entities appearing in the query. As a result, previous approaches are inadequate in addressing the factual hallucination appearing in the reasoning processes of LLMs.

In this paper, we propose Knowledge Graph-based Retrofitting (KGR), a new framework that incorporates LLMs with KGs to mitigate factual hallucination during the entire reasoning process of LLMs. Instead of retrieving factual information from KGs using original queries, the main idea behind KGR is to autonomously retrofit the initial draft responses of LLMs based on the factual knowledge stored in KGs. However, achieving the above process is challenging because draft responses generated by large language models typically contain a mixture of various information about the reasoning process, making the extraction, verification, and revision of relevant knowledge in it very challenging. Therefore, the key to integrating Knowledge Graphs into the reasoning process of large models to mitigate factual hallucinations lies in efficiently extracting the information requiring
Figure 1: An overview of KGR, our framework consists of five components: (1) claim extraction, (2) entity detection and KG retrieval, (3) fact selection, (4) claim verification, (5) response retrofitting. The core component of these five steps remains the LLM. Given a question and draft response as input, our framework can iteratively mitigate factual errors in LLM’s response.

In summary, the contributions are as follows:

- We propose a new framework that incorporates LLMs with KGs to mitigate factual hallucination by effectively extracting, verifying, and refining factual knowledge in the entire reasoning process of LLMs.
- We present an implementation of the above-mentioned procedure by executing all the above-mentioned steps using LLMs without introducing any additional efforts.
- Experiments on 3 datasets and 3 different LLMs confirm that KGR can significantly mitigate the hallucination and enhance the reliability of LLMs.

Related Work

Hallucination Hallucination in Large Language Models has been a prominent research focus within the NLP community (Ji et al. 2023). Automated large-scale data collection processes are prone to collecting erroneous information, which can significantly impact the quality of the generated outputs (Gunasekar et al. 2023). Additionally, excessive repetition of certain data during training can introduce memory biases, further exacerbating the hallucination issue (Lee et al. 2022; Biderman et al. 2023). Imperfections in the encoder backbone and variations in decoding strategies also play a role in determining the extent of hallucination in LLMs outputs (Tian et al. 2019). Recent studies have emphasized the importance of model output confidence as an indicator of potential hallucination occurrences (Manakul, Liusie, and Gales 2023).

Retrieval Augmentation To address hallucination issues in LLMs, two main categories of retrieval augmentation methods have been proposed, which can be concluded as “retrieve before generation” and “retrieve after generation”. The retrieve before generation mainly focuses on leveraging information retrieval (IR) to provide additional information to LLMs about the query. Along this line, UniWeb (Li et al. 2023b) introduces an adaptive method for determining the optimal quantity of referenced web text, Chameleon (Lu et al. 2023) leverages an assortment of tools including search engines, to bolster the reasoning capabilities of LLMs, We-
bGLM (Liu et al. 2023b) augments LLMs with web search and retrieval capabilities. One major limitation of these approaches is the retrieved text is question-related, thus cannot guarantee the correctness of the question-unrelated portions in the generations. The retrieve after generation like RARR (Gao et al. 2023), PURR (Chen et al. 2023), and CRITIC (Gou et al. 2023) automatically edit model generations using evidence from the web. Our method leverages KGs as knowledge base to retrofit the model-generated response while reducing hallucination risk.

**KG-Enhanced LLM** The Knowledge Graph is regarded as a dependable source of information and is consequently frequently employed to enhance model generations. Traditional approaches involve knowledge representations during the training phase, which often necessitates dedicated model architecture and model-specific training (Zhang et al. 2019, 2022). However, this incurs a substantial cost for contemporary LLMs. In recent years, many researchers propose to inject knowledge during the inference stage. For example, KAPING (Baek, Aji, and Saffari 2023), RHO (Ji et al. 2022), KITLM (Agarwal et al. 2023), and StructGPT (Jiang et al. 2023) try to retrieve knowledge in KG and utilize them as an additional input context for LLMs to enhance their generations. However, these methods only search for question-relevant information, which limits the overall performance. To the best of our knowledge, we’re the first to involve knowledge graphs in model response retrofitting.

**KGR: Autonomous Knowledge Graph-Based Retrofitting**

In this section, we introduce our proposed method KGR, which automatically mitigates factual hallucinations via a chain-of-verification process. As shown in Figure 1, given a query and its draft response, KGR retrofits the response by 1) extracting claims from the draft response that requires verification; 2) detecting entities in the claims that are critical for retrieving facts from knowledge graph; 3) retrieving relevant fact statements from the knowledge graph; 4) verifying the factual correctness of each extracted claim using the returned fact statements from the knowledge graph; 5) retrofitting the previous draft response based on the verification results. All these steps are autonomously executed using the large language model itself without additional manual efforts. This process can be iterative and repeated multiple times to ensure that all facts in the generated answers align with the factual knowledge stored within the knowledge graph. In the following, we will describe each component in KGR respectively in detail.

**Claim Extraction**

Given a generated draft response as input, claim extraction will extract all factual claims from previously generated drafts that require validation. The main idea behind claim extraction is that a draft response can frequently contain various factual statements that need to be verified. For the example in Figure 1, the draft response contains at least two factual statements, i.e., “Frédéric Chopin’s father is Nicolas Chopin” and “Nicolas Chopin was born on June 17, 1771”. Therefore, to make it possible for KG to verify these statements respectively, claim extraction decomposes the draft response to be atomic factual claims.

In this paper, we leverage LLM itself to autonomously extract the claims in the generated draft response. As shown in Figure 2, we prompt LLM with a query and response pair, with the anticipation of receiving a list of decomposed factual claims.

After extracting claims, entity detection identifies the mentioned critical entities for knowledge graph retrieval.

**Entity Detection and Knowledge Graph Retrieval**

In this paper, we leverage LLM itself to autonomously extract the claims in the generated draft response. As shown in Figure 2, we prompt LLM with a query and response pair, with the anticipation of receiving a list of decomposed factual claims.

After extracting claims, entity detection identifies the mentioned critical entities for knowledge graph retrieval.

**Fact Selection**

Given a list of claims extracted from the draft response, entity detection will detect the critical entities mentioned in the claims. Then, we retrieve the detected entities’ local subgraph from the KG and express it in the form of triples. The main idea behind entity detection and knowledge graph retrieval is that we need to identify entities in claims so as to retrieve the relevant knowledge in the KG. Meanwhile, we can ensure recalling relevant triples as much as possible by retrieving the local subgraph in the knowledge graph. For the example in Figure 1, we identify the entity Frédéric Chopin and its entity id Q1268, so we can search the identified entity to acquire knowledge relevant to Claim1 in the KG.

In this paper, we prompt LLMs to detect entities. As illustrated in Figure 3, our approach shows powerful generalization ability by capitalizing on the information extraction capabilities of LLMs(Li et al. 2023a) through the utilization of few-shot prompt. Based on the few-shot prompts, we can make LLMs understand which entities merit fact selection. After detecting the entities, we retrieve the knowledge graph for the local subgraph and send it to fact selection in the form of triples. Concretely, we retrieve the node identifiers for these entities in Wikidata using heuristic rules and then query the knowledge graph for the neighboring nodes of the identified entity and obtain its local subgraph with SPARQL.
In this paper, we partition retrieved triples into several chunks and leverage LLM itself to extract the critical triples in the retrieved triples respectively, illustrated in Figure 4. In this way, we can avoid introducing excessive irrelevant knowledge into claim verification. Once we have selected the critical triples, the claim verification will verify the factual correctness of claims and subsequently offer suggestions.

Claim Verification

Given the critical triples selected by the fact selection, we utilize LLM to compare the model-generated claims with the factual information present in the KGs. The main idea behind claim verification is to propose a detailed revision suggestion for each claim, as retrofitting solely based on the selected knowledge may not convince LLMs. As illustrated in Figure 5, we employ LLMs to verify each claim and propose revision suggestions respectively based on the retrieved fact knowledge, so as to boost the execution of the following retrofitting step. Then, we send the claim verification result to LLM to ask it to retrofit the draft response accordingly.

Response Retrofitting

Given the verification of all claims, the response retrofitting step retrofits the generated draft response in accordance with the verification suggestions.

In this paper, we capitalize on the capabilities of LLMs for the purpose of retrofitting. This approach involves employing LLMs with a few-shot prompt, a strategy that has exhibited efficacy in prior researches (Gou et al. 2023; Zheng et al. 2023). As illustrated in Figure 5, we merge the entire KGR process into a singular prompt. This allows LLMs to leverage their in-context learning ability, comprehending the KGR process and enhancing their comprehension of factual retrofitting based on verification suggestions.

By following the cycle of “Extraction - Detection - Selection - Verification - Retrofitting”, our KGR framework can be iterated multiple times to ensure all facts in the generated answers align with the factual knowledge stored within the knowledge graph.

Experiments

We evaluate our KGR framework on three datasets with different levels of reasoning difficulty, including Simple Question (Bordes et al. 2015), Mintaka (Sen, Aji, and Saffari 2022), and HotpotQA (Yang et al. 2018). We also compare KGR with information retrieval-based approaches and previous question-relevant knowledge graph retrieval approaches.

Experiment Settings

Dataset and Evaluation We conduct experiments on three representative factual QA benchmarks, including:

- **Simple Question (Bordes et al. 2015)** is a simple QA dataset that contains 100k questions constructed from Freebase knowledge graph, requiring no deep reasoning procedure. Therefore, we can evaluate the ability to retrieve relevant evidence in KG based on Simple Question.
- **Mintaka (Sen, Aji, and Saffari 2022)** is a complex, natural and multilingual dataset, composing 20k questions collected in 8 different languages. We only use English
**Simple Question Mintaka HotpotQA**

<table>
<thead>
<tr>
<th></th>
<th>ChatGPT</th>
<th>text-davinci-003</th>
<th>ChatGPT</th>
<th>text-davinci-003</th>
<th>ChatGPT</th>
<th>text-davinci-003</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla</td>
<td>22.0/28.9</td>
<td>34.7/45.1</td>
<td>42.9/56.1</td>
<td>36.7/44.8</td>
<td>18.4/31.6</td>
<td>22.4/34.6</td>
</tr>
<tr>
<td>CoT</td>
<td>10.0/11.8</td>
<td>38.0/46.7</td>
<td>53.1/59.3</td>
<td>46.3/57.9</td>
<td>24.5/34.3</td>
<td>29.2/40.5</td>
</tr>
<tr>
<td>CRITIC</td>
<td>12.0/14.3</td>
<td>38.0/46.7</td>
<td>51.0/58.6</td>
<td>44.4/54.0</td>
<td>30.6/41.7</td>
<td>27.1/38.9</td>
</tr>
<tr>
<td>QKR</td>
<td>54.0/60.2</td>
<td>56.0/62.0</td>
<td>48.0/54.6</td>
<td>44.0/51.7</td>
<td>28.0/38.1</td>
<td>22.0/31.9</td>
</tr>
<tr>
<td>KGR (ours)</td>
<td>58.0/60.7</td>
<td>54.0/57.6</td>
<td>53.1/60.8</td>
<td>52.0/60.2</td>
<td>32.7/39.2</td>
<td>34.0/47.2</td>
</tr>
</tbody>
</table>

Table 1: Results on three datasets using ChatGPT and text-davinci-003. We implement CoT using the prompt provided by CRITIC. QKR uses the same entity detection and fact selection method as KGR. We report both EM and F1 scores in the table.

<table>
<thead>
<tr>
<th>Simple Question</th>
<th>Mintaka</th>
<th>HotpotQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoT</td>
<td>14.0/21.9</td>
<td>26.0/28.3</td>
</tr>
<tr>
<td>QKR</td>
<td>40.0/44.0</td>
<td>26.5/32.4</td>
</tr>
<tr>
<td>KGR (ours)</td>
<td>46.0/46.9</td>
<td>26.5/34.0</td>
</tr>
</tbody>
</table>

Table 2: Results on three datasets using Vicuna 13B. We report both EM and F1 scores in the table.

**LLMs and KG Implementation** We evaluate the effectiveness of KGR on both close-source and open-source large language models. For close-source models, we evaluate on text-davinci-003 and ChatGPT (gpt-3.5-turbo-0301) to see whether alignment will have an impact on KGR. For the open-source model, we evaluate KGR on Vicuna 13B, a representative aligned open-source model, to see whether KGR can work well on compact size LMs. We choose Wikidata\(^1\) (Vrandečić and Krötzsch 2014) as our knowledge base, which encompasses structured data from various sources such as Wikipedia, Wikimedia Commons\(^2\) (Commons 2023), and other wikis associated with the Wikimedia movement\(^2\) (Meta 2023).

**Overall Results**

As shown in Table 1, our method demonstrates significant superiority over other methods across various conditions.

1) **Our framework can mitigate large language model hallucination via Knowledge Graph-based Retrofitting and achieve significant improvements on 3 datasets.** Compared with the CoT and CRITIC, our KGR framework gains improvements on all three datasets. This indicates that our KG-based approach is more effective due to its reliance on a reliable knowledge base, whereas IR-based methods like CRITIC might introduce noise from external. Additionally, we observed that the CoT

---

\(^1\)https://wikipedia.org

\(^2\)https://www.wikidata.org/

\(^3\)https://www.bing.com/.
method performed worse than the vanilla approach in ChatGPT. This could be attributed to the CoT method’s tendency to ask for more information, which is amplified in ChatGPT due to Reinforcement Learning from Human Feedback (Ouyang et al. 2022).

2) By verifying the facts used during reasoning via chain-of-verification, our method can achieve significant performance improvement in complex reasoning tasks in Mintaka and HotpotQA datasets. As shown in Table 1, compared to the QKR method, our KGR framework achieves F1 improvement for at least 6.2 and 1.1 on Mintaka and HotpotQA. Both of them pose complex reasoning question-answering challenges, and the success of our method with chain-of-verification on these datasets demonstrates its capability to handle complex questions effectively. It is worth noting that the text-davinci-003 outperformed QKR in Simple Question. We attribute this to the fact that Simple Question consists of straightforward, one-hop questions, which makes the question-relevant method more effective.

3) By automatically generating and executing chain-of-verification via LLMs, our KGR approach exhibits remarkable generalization capabilities in different datasets and is robust on open-domain settings. In HotpotQA, KGR performs well compared to the CoT and CRITIC methods. The HotpotQA presents an open-domain QA scenario where finding related triples in the KG can be challenging. Despite this difficulty, our method displayed the ability to effectively utilize the searched triples and effectively leverage parametric knowledge even when no evidence was returned.

4) Our framework can work well on compact size LMs, aligned LLMs, and misaligned LLMs, showing the generalizability of KGR. We compare KGR with the strong baselines CoT and QKR on Simple Question, Mintaka, and HotpotQA using Vicuna 13B. The result is shown in Table 2. We can find that the KGR framework outperforms both CoT and QKR, demonstrating the generalizability of our framework even leveraging a compact size LM. Moreover, the significant improvement with ChatGPT and text-davinci-003 shows the generalizability of both aligned LLMs and misaligned LLMs.

In summary, our method consistently outperforms other methods across various conditions and exhibits strong generalization ability. The results suggest that our KGR framework is more reliable and effective, especially in handling complex factual reasoning tasks. Furthermore, it showcases the robustness of our method in open-domain QA settings, where knowledge retrieval may be more challenging.

**Case Study**

We present a multi-round retrofitting process of a multi-hop case which needs to be retrofitted iteratively in Figure 3. In this case, the model-generated response shows a factual error in the initial reasoning step. It erroneously states that **Alex Shevelev died in Moscow, Russia**, whereas he actually passed away in Rome, Italy. After retrofitting this mistake, we encounter another factual error, which asserts that **Rome** is the capital of the Central Federal District. So, we need to retrofit it again based on the retrofitted response in the first iteration.

From this case, we show KGR’s intermediate results, including atomic claim, critical triples, detailed verification, and iterative retrofitting. All these show the effectiveness of KGR, especially on reasoning with multi-hop complex tasks, verifying the feasibility of multi-turn retrofit to ensure that all facts in the generated answers align with the factual knowledge stored in the knowledge graph.

**Error Analysis**

In order to gain a comprehensive understanding of the KGR approach, we conducted an exhaustive analysis of incorrect cases based on the Mintaka and Simple Question datasets. After carefully examining the errors, we identified which component causes revision failures. The outcomes of this analysis are visualized in Figure 6. On closer inspection, the main issues are inaccuracies in entity detection and fact selection, while claim extraction, verification, and response retrofitting are more reliable, underscoring the need to improve entity detection and fact selection.

On the other hand, our analysis explored the error reason in different stages. The claim extraction often fails to express the central claim adequately, sometimes due to excessive use of pronouns that confuse the model’s comprehension. Entity detection has issues with entity extraction granularity. It captures too many common entities like "films" or "apple", leading to excessive and useless triples for the claim verification. The fact selection has difficulty extracting the critical triples between multiple triples that contain noise. For the claim verification and response retrofitting, the focus shifts to the model’s ability to adhere to the cues provided by the few-shot prompts. Effectively discerning and subsequently rectifying answers within this framework presents a central challenge. The process of fact selection encounters challenges in extracting essential triples from a collection of triples that include irrelevant information or noise.
Triples with triples in random orders doesn’t significantly affect triple selection. As discussed above, considering the limitation of maximum size increases the chance of selecting both critical and irrelevant triples. Additionally, we observe that prompting LLMs using ChatGPT. These experiments help us understand fact selection behavior under various hyperparameters, optimize chunk size, and refine triple retrieval strategies for improved efficiency.

As shown in Figure 7 a), the chunk size has minimal impact on triple selection capability, except for a chunk size of 100, which may cause worse long-distance dependency modeling. However, reducing the chunk size leads to lower precision and higher recall scores. This indicates that a smaller chunk size increases the chance of selecting both critical and irrelevant triples. Additionally, we observe that prompting LLMs with triples in random orders doesn’t significantly affect triple selection.

As shown in Figure 7 b), increasing the number of retrieved triples has a gradual positive impact on recall but significantly reduces precision. More retrieved triples may boost recall for critical knowledge and introduce numerous irrelevant triples, potentially compromising the effectiveness of the claim verification and negating the benefits of fact selection.

All in all, experiments show that the core difficulty of retrieving fact knowledge based on LLMs is the tradeoff between precision and recall. This observation points to future research on fact selection based on LLMs.

### Impact of Chunk Size&Numbers of Retrieved Triples

As discussed above, considering the limitation of maximum input length for LLMs, we partition the retrieved triples into chunks for fact selection. We evaluate the effectiveness of fact selection when retrieved triples are in random order, referring black point in Figure 7 a) on the Simple Question using ChatGPT. These experiments help us understand fact selection behavior under various hyperparameters, optimize chunk size, and refine triple retrieval strategies for improved efficiency.

As shown in Figure 7 a), the chunk size has minimal impact on triple selection capability, except for a chunk size of 100, which may cause worse long-distance dependency modeling. However, reducing the chunk size leads to lower precision and higher recall scores. This indicates that a smaller chunk size increases the chance of selecting both critical and irrelevant triples. Additionally, we observe that prompting LLMs with triples in random orders doesn’t significantly affect triple selection.

### Conclusion

In this paper, we propose a knowledge graph-based retrofitting framework that effectively mitigates factual hallucination during the reasoning process of LLMs based on the factual knowledge stored in KGs. Experiment results show that KGR can significantly improve the performance of LLMs on factual QA benchmarks especially when involving complex reasoning, which demonstrates the necessity and effectiveness of KGR in mitigating hallucination and enhancing the reliability of LLMs. As for future work, we plan to improve the effectiveness in each step of our KGR framework.

### Acknowledgements

We thank all reviewers for their valuable suggestions. This research work is supported by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant No. XDA27020200, the National Natural Science Foundation of China under Grants no. 62122077, 62106251, 62306303 and 62076233.

### References


Chase, H. 2022. LangChain.


Commons, W. 2023. Main Page — Wikimedia Commons, the free media repository.


