Zero-resource Hallucination Detection for Text Generation via Graph-based Contextual Knowledge Triples Modeling

Xinyue Fang[1], Zhen Huang[1*], Zhiliang Tian[1*], Minghui Fang[2], Ziyi Pan[1], Quntian Fang[1], Zhihua Wen[1], Hengyue Pan[1], Dongsheng Li[1]

[1]College of Computer, National University of Defense Technology
[2]College of Computer Science and Technology, Zhejiang University
{fangxinyue, huangzhen, tianzhiliang, panziyi, fangquntian, zhwen, hengyuepan, dsli}@nudt.edu.cn
minghuifang@zju.edu.cn

Abstract

LLMs obtain remarkable performance but suffer from hallucinations. Most research on detecting hallucination focuses on questions with short and concrete correct answers that are easy to check faithfulness. Hallucination detections for text generation with open-ended answers are more hard. Some researchers use external knowledge to detect hallucinations in generated texts, but external resources for specific scenarios are hard to access. Recent studies on detecting hallucinations in long texts without external resources conduct consistency comparison among multiple sampled outputs. To handle long texts, researchers split long texts into multiple facts and individually compare the consistency of each pair of facts. However, these methods (1) hardly achieve alignment among multiple facts; (2) overlook dependencies between multiple contextual facts. In this paper, we propose a graph-based context-aware (GCA) hallucination detection method for text generations, which aligns facts and considers the dependencies between contextual facts in consistency comparison. Particularly, to align multiple facts, we conduct a triple-oriented response segmentation to extract multiple knowledge triples. To model dependencies among contextual triples (facts), we construct contextual triples into a graph and enhance triples' interactions via message passing and aggregating via RGCN. To avoid the omission of knowledge triples in long texts, we conduct an LLM-based reverse verification by reconstructing the knowledge triples. Experiments show that our model enhances hallucination detection and excels all baselines.

Code —
https://github.com/GCA-hallucinationdetection/GCA
Technical Appendices — https://arxiv.org/abs/2409.11283

1 Introduction

Recent research has shown that large language models (LLMs) achieved state-of-the-art performance in various NLP tasks (Fang et al. 2024; Lu et al. 2022). However, these models often suffer from hallucinations: generate incorrect or fabricated content in a factual way, which undermines models' credibility (Ji et al. 2023; Lu et al. 2023a) and limits LLMs' application in fields requiring factual accuracy (Huang et al. 2023; Su et al. 2024). Detecting hallucination in the model's responses is crucial for LLMs' boom.

Existing hallucination detection studies mainly concentrate on tasks like question answering (QA) (Zhang et al. 2023a; Wen et al. 2023a) and arithmetic calculation (Xue et al. 2023; Tian et al. 2022) with short and concrete correct answers. In these tasks, the consistency among their concrete answers can be easily checked for hallucination (Jiang et al. 2021; Wang et al. 2024a). The research on detecting hallucinations in generated long text is more challenging because (1) generating text is open-ended and rarely has concrete answers and (2) long text contains multiple facts and the consistency among them is hard to verify. Therefore, we focus on hallucination detection in long text generation with black-box models (powerful LLM, like GPT-4) without external resources (i.e. zero-resource setting).

Currently, the studies on black-box zero-resource hallucination detection for text generation can be divided into two categories: (1) **Self-checking** (Friel and Sanyal 2023; Liu et al. 2023) designs prompt texts using chain-of-thought (CoT) to verify response factuality by the LLMs' own capabilities. Though that can be easily implemented and applied in various scenarios, it relies on the model's own ability and can lead to missed detection: The model may overconfidently trust its outputs (Li et al. 2024a; Chen et al. 2024). (2) **Consistency comparison** (Zhang et al. 2023a; Ma et al. 2024) samples multiple responses to check whether the sampled responses are highly inconsistent, which indicates hallucination (Farquhar et al. 2024; Wang et al. 2024b). The method is effective for short responses with few concrete claims, making consistency comparison easy. However, in long text responses, diverse wording or lexical representations of the same fact complicate consistency comparison.

To address those issues, researchers propose **divide-and-conquer** (Zhang et al. 2024) based on consistency comparison. It has three steps: (1) sample multiple additional responses appending to the original response; (2) divide each response into multiple facts; (3) compare the consistency of facts in the original response with those in the sampled responses, where highly inconsistent facts indicate hallucinations. However, it ignores the omission issue: facts in the original but not in sampled responses may be wrongly seen as hallucinations for lack of comparison. To overcome that, (Yang, Sun, and Wan 2023; Cao, Yang, and Zhao 2024) propose **reverse verification**. For each fact in the original response, they prompt LLMs to create a new question

*Corresponding Author

answered by the claim and check if the model's response matches the fact. The method can avoid the omission issue because each fact has an accurately matched counterpart for comparison. However, a long text may contain multiple facts $\{A_1, ...A_N\}$, where A_i is the i-th fact within LLMs' generation. The above method compares the consistency of each fact sampled from different responses individually (e.g. verify the truthfulness of A_i by comparing with B_i, where B_i is A_i's corresponding fact from a sampled response). It ignores the dependencies among each fact that can reduce detection performance. For example, for the response "Einstein won the Nobel Prize for proposing the theory of relativity." we can extract two triples: (Einstein, proposal, theory of relativity) and (Einstein, receive, Nobel Prize). Although each triple is a fact, their dependency is an error. Therefore, considering the dependencies between a fact and its surrounding facts is promising to enhance hallucination detection.

In this paper, we propose a **graph-based context-aware (GCA)** hallucination detection method for long text generation, which extracts triples to align facts for a better consistency comparison and considers the dependencies between contextual triples. Specifically, to better align facts from responses for consistency comparison, we conduct a triple-oriented response segmentation to extract multiple knowledge triples for each response. Triples are well-structured and thus easy for comparison; in addition, it is easy to align facts among multiple responses since triples carry only facts and get rid of the wording and lexical representations in the response (Wang et al. 2023b) (Sec. 3.2). To consider dependencies between a triple and its contextual triples, we construct a graph over a triple and its contextual triples. Then, we encourage triples' interactions via message passing and aggregating features of neighboring nodes via RGCN (relational graph convolutional network) (Schlichtkrull et al. 2018) (Sec. 3.3). It encourages the interaction between facts as we observe that a certain number of facts has dependencies in a response (See Sec. 4.5). To avoid the omission issue in (Sec. 3.3), we propose an LLM-based reverse verification method with three reconstruction tasks to reconstruct the knowledge triples (Sec. 3.4). These tasks provide a more thorough and detailed detection of each triple in long text responses, enhancing the overall effectiveness of the method. Experiments show that our method effectively improves hallucination detection accuracy for long text responses generated by black-box models under zero-resource conditions.

Our contributions are: (1) we propose a hallucination detection method for long text generation that considers the dependencies between contextual knowledge triples. (2) We propose a graph-based context-aware hallucination detection via consistency comparison with RGCN. (3) We additionally proposed three reversion verification tasks to help hallucination detection by reconstructing triples. (4) Experiments show that our method outperforms all baselines.

2 Related Work

2.1 White-box Hallucination Detection

These methods analyze the model's internal states to identify hallucinations (Yuksekgonul et al. 2024; Lu et al. 2023b; Wen et al. 2024), mainly divided into two types: (1) *Output logit based method:* The model's output logits reflect the confidence of the model's predictions (Jiang et al. 2024). (Varshney et al. 2023) calculates the logits for concepts in response and takes the minimal probabilities to model the uncertainty. (Verma et al. 2023) explores the integration of Epistemic Neural Networks (ENNs) with LLMs to improve the model's uncertainty estimates. (Luo, Xiao, and Ma 2023) proposes to adjust the model's output logits by adding a linear layer to better align with correctness likelihood. (2) *Hidden layer activations based method:* Hidden layer activations encapsulate the model's internal representation of statement truthfulness (Fadeeva et al. 2024). (Azaria and Mitchell 2023) trains a classifier using LLM's hidden layer activations to assess the truthfulness of statements. (Snyder, Moisescu, and Zafar 2024) uses output values from artifacts associated with model generation as input features to train a classifier that identifies hallucinations. (Zhu et al. 2024) uses probabilistic models to analyze internal state transitions in the LLM during generation to detect hallucinations.

2.2 Black-box Hallucination Detection using External Resources

These methods aim to verify the authenticity of model-generated content by leveraging external knowledge (Wen et al. 2023b; Nahar et al. 2024). Depending on the source of external knowledge, it can be categorized into the following two types. (1) *RAG-based method:* Retrieval-augmented generation (RAG) is a technique that enhances text generation by retrieving relevant information from external sources (Sadat et al. 2023; Wang et al. 2023a). (Roychowdhury et al. 2024) proposes a multi-LLM system with the capability to retrieve external knowledge bases and perform real-time content authentication. (Ding et al. 2024) retrieves relevant evidence to help LLMs correct potential hallucinations in responses. (Kang, Ni, and Yao 2024) proposes a method for the real-time retrieval of Web search engines that can verify the factuality of the output responses and correct hallucinations. (Li et al. 2024b) automatically retrieves knowledge graphs to detect hallucinations through logical programming and mutation testing. Furthermore, (Bayat et al. 2023) proposes an automated method to extract factual claims from responses and collect evidence from knowledge graphs to verify the factuality of the claims to be extracted. (2) *Incorporating Alternative Models:* Researchers use responses generated by other models for cross-validation to detect hallucinations (Hegselmann et al. 2024). (Cohen et al. 2023) constructs a framework for assessing the factuality of output responses through cross-validation by two language models. (Rawte et al. 2024; Wan et al. 2024) use multiple LLMs as "judges" to evaluate various aspects of the model's output responses. (Li et al. 2024b) proposes a mutation testing model based on logic programming, which can verify the consistency of LLMs' responses with real-world situations.

2.3 Black-box Hallucination Detection using Zero-resource

Researchers propose using the model's own capabilities to detect hallucinations (Liu et al. 2024) because obtain-

ing high-quality external resources is challenging (Mündler et al. 2024). (1) *For non-long text responses generated by the model:* Consistency comparison through multiple sampling responses is an important method (Allen, Polat, and Groth 2024). (Zhang et al. 2023a) improves hallucination detection performance on commonsense QA tasks through semantic-aware cross-validation consistency. (Liu et al. 2024) evaluates the reliability of responses generated by LLMs for individual questions or queries through cross-query comparison. (Ma et al. 2024) proposes a critical calculation and conclusion (CCC) prompt template to enhance LLM's ability to detect and correct unreasonable errors in mathematical problem-solving. (Yehuda et al. 2024) identifies potential instances of hallucination by quantifying the level of inconsistency between the original query and the reconstructed query. (2) *For long text responses generated by the model:* (Manakul, Liusie, and Gales 2023) proposes a method to detect hallucinations by comparing the consistency of responses from multiple random samplings. (Yang, Sun, and Wan 2023) introduces a reverse validation method for passage-level hallucination detection in LLMs, which leverages the LLM's own knowledge base. (Mündler et al. 2024) introduces a method for detecting self-contradictions in long text responses through logical reasoning. (Friel and Sanyal 2023) proposes an efficient prompting method that uses the chains of thought generated by LLMs to detect hallucinations in the responses. Unlike these LLM-based methods, our approach constructs long text responses as graphs and uses graph neural networks to capture the contextual influence of each fact during hallucination detection.

3 Method

3.1 Overview

Our method has three modules: (1) **Triple-Oriented Response Segmentation** (Sec. 3.2) extracts facts from the model's responses. (2) **Graph-based Contextual Consistency Comparison with RGCN** (Sec. 3.3) constructs a graph carrying the extracted knowledge triples and utilizes an RGCN to propagate and integrate messages across the graph. It considers the dependencies between each knowledge triple (facts) and its surrounding triples during detection. (3) **Reverse Verification via Triples Reconstruction** (Sec. 3.4) achieves reverse verification for hallucination detection by reconstructing each triple via three LLM-based tasks (as shown in Fig.1).

We feed each knowledge triple extracted (Sec. 3.2) to detect hallucinations (Sec. 3.3 and Sec. 3.4), and then we judge the original long text response relying on the results of each triple from (Sec. 3.3 and Sec. 3.4).

3.2 Triple-Oriented Response Segmentation

To better align facts in the consistency comparison, we propose to segment the responses by extracting knowledge triples as facts and checking the answers' consistency among the triples. Our motivation is that due to the impact of wording, comparing textual consistency can lead to mismatches. Because hallucination detection considers the semantics of knowledge fact instead of specific word choices, we use a triple-based comparison method to provide better alignment than traditional textual comparison. Specifically, the steps are as follows:

- **Extraction.** Inspired by the latest method (Hu et al. 2024), we design prompts to extract knowledge triples from responses using an LLM.
- **Verification.** To ensure the accuracy of the extracted knowledge triples, we pair each response with its triples and prompt the LLM to confirm their semantic equivalence. If any ambiguities exist between the extracted triples and the response text, we instruct the LLM to adjust the semantics of the triples according to the text's semantics. The details of the prompts are in App.A.

Knowledge triples have a structured format and are easy to compare, simplifying alignment and comparing consistency between responses, enhancing detection accuracy.

3.3 Graph-based Contextual Consistency Comparison with RGCN

To effectively consider dependencies between triples, we propose Graph-based Contextual Consistency Comparison **(GCCC)**, which constructs a knowledge graph for each response and then conducts message passing and aggregation via RGCN. The intuition is that traditional consistency comparison focuses on comparing individual facts: it verifies a single piece of knowledge fact by comparing it only with the corresponding fact in the sampled responses at a time. It results in ignoring the triples that are mutually dependent on the given triple within the context information.

To address the problem, our approach constructs graphs for the original response and the sampled responses. Then, it employs RGCN for message passing and aggregation on these graphs. The process consists of two stages: (1) *knowledge triples modeling via graph learning*. We build a graph for each response and obtain node (entity) embeddings via RGCN processing to model the multiple knowledge triples for a response. (2) *triples consistency comparison*. We compare the consistency of triples across the graphs at the embedding level.

Knowledge Triples Modeling via Graph Learning This stage is divided into three steps: firstly, we convert each response into a graph. Then, we obtain the initial node (entity) embeddings for each graph using sentence-BERT (Wang et al. 2020). Finally, we employ the RGCN to perform message passing and aggregation using the initial node embeddings on each graph, updating the node embeddings.

- **Graph Construction.** For a user's query, the model generates an original response R_{o}, and we sample multiple additional responses $R_{\text{sampled}} = \{R_1, R_2, \ldots, R_n\}$. (h_i, r_i, t_i) is a single triple in R_{o} and $(h_{i,j}, r_{i,j}, t_{i,j})$ is a single triple in j-th sampled response R_j. We construct the graph $G_{\text{o}} = (V_{\text{o}}, E_{\text{o}})$ for the original response, in which vertices $(v \in V_{\text{o}})$ represent the head and tail entities from each triple. An edge $(e \in E_{\text{o}})$ represents the relation between the head entity and the tail entity. Similarly, we construct the graph $G_j = (V_j, E_j)$ for each

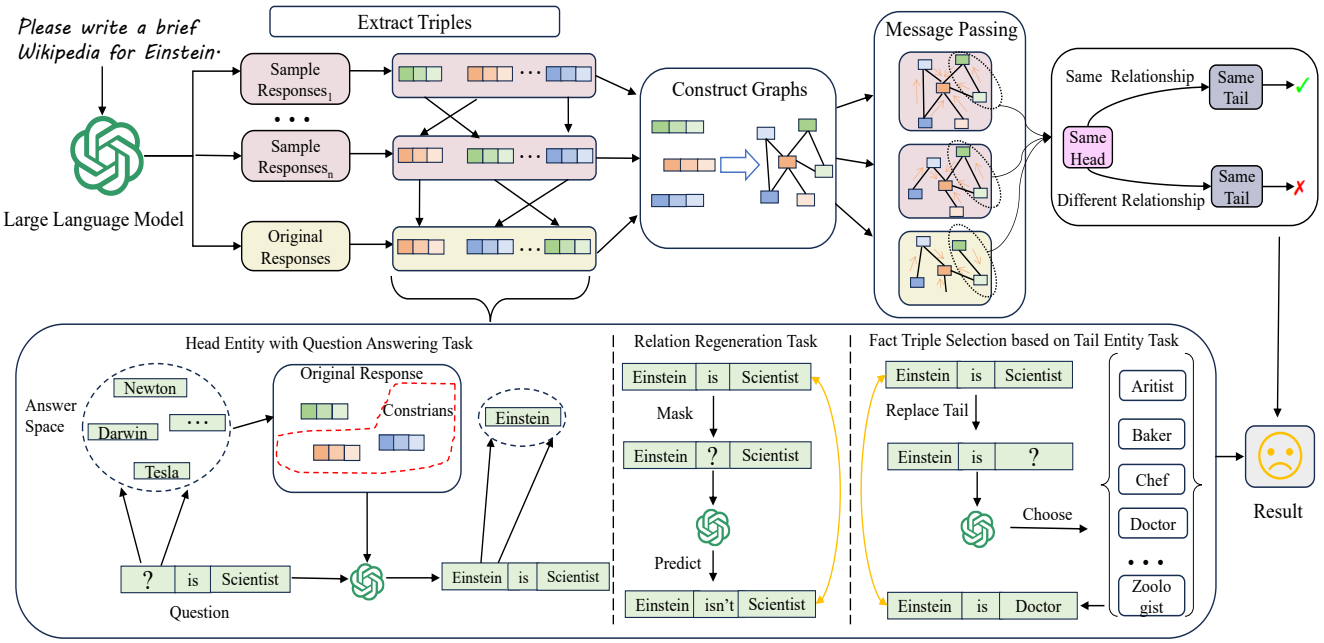

Figure 1: GCA framework. We extract triples from the original response and sampled responses (left-upper corner). Then, we construct a graph for each response with the extracted triples and perform message passing and aggregation on the graph (as the upper branch). We conduct reverse validation for each part of the triples with three reconstruction tasks (as the lower branch).

sampled response R_j. By doing so, we construct several response-specific graphs for each user's query.

- **Representation Initialization.** Using sentence-BERT, we encode the head entities, relation, and tail entities in knowledge triples as vector representations. For the original response, we represent each triple embedding as: $(\mathbf{h}_i, \mathbf{r}_i, \mathbf{t}_i) = \text{BERT}(h_i, r_i, t_i)$. For each sampled response, we represent each triple embedding as: $(\mathbf{h}_{i,j}, \mathbf{r}_{i,j}, \mathbf{t}_{i,j}) = \text{BERT}(h_{i,j}, r_{i,j}, t_{i,j})$. We treat the head and tail entity embeddings from the original response as G_o's initial node (entity) embeddings. Similarly, we obtain the initial node (entity) embeddings for the graph G_j corresponding to j-th sampled response.
- **Message Passing and Aggregation.** We use the RGCN to perform message passing and aggregation on the graph. As Eq.1 shows that for each layer l, the new representation of each node v is denoted as $\mathbf{e}_v^{(l+1)}$. For each relation $r \in \mathcal{R}$, we denote the set of all neighbors of the node v that are connected through an edge of relation r as $\mathcal{N}_r(v)$. For each neighbor in $\mathcal{N}_r(v)$, we multiply its representation $\mathbf{e}_u^{(l)}$ by a weight matrix $\mathbf{W}_r^{(l)}$ and normalize it using the hyperparameter $c_{v,r}$. In addition to aggregating information from neighbors, $\mathbf{e}_v^{(l+1)}$ also includes its own representation $\mathbf{e}_v^{(l)}$ from the previous layer l and transform it by a weight matrix $\mathbf{W}_0^{(l)}$.

$$\mathbf{e}_v^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{N}_r(v)}\frac{1}{c_{v,r}}\mathbf{W}_r^{(l)}\mathbf{e}_u^{(l)} + \mathbf{W}_0^{(l)}\mathbf{e}_v^{(l)}\right) \quad (1)$$

The updating for $\mathbf{e}_v^{(l+1)}$ integrates information from v's neighbors through relation-specific change, while also incorporating v's own representation. These operations ensure that the updated node embedding is informed by its context and intrinsic properties. Triples containing the node can also incorporate contextual information, enhancing the accuracy when comparing the consistency of triples, thereby improving the detection of hallucinations.

Triples Consistency Comparison. Based on the graph representations from RGCN, we detect hallucinations in the original response by comparing the consistency of triples across multiple graphs. Firstly, we align triples between the original response's graph G_o and each sampled graph G_j. Then we compare the consistency of the aligned triples to calculate the consistency score.

- **Triples Alignment.** For each triple (h_i, r_i, t_i) in the original response and each triple $(h_{i,j}, r_{i,j}, t_{i,j})$ in the sampled response, we first check whether the head entities of these two triples are the same. If so, we calculate the similarity $S(\mathbf{r}_i, \mathbf{r}_{i,j})$ between the relation representation $\mathbf{r}_i$ of relation r_i and the representation $\mathbf{r}_{i,j}$ of $r_{i,j}$. If $S(\mathbf{r}_i, \mathbf{r}_{i,j})$ exceeds the pre-defined threshold θ_r, we regard the two triples as aligned. Otherwise, they are considered unaligned. For every triple in the original response, we apply the above operations to align each triple from sampled responses with it.
- **Consistency score calculation.** After aligning the triples, we need to further compare whether they are consistent with each other to calculate the consistency score.

Specifically, as Eq. 2 shows, for a triple (h_i, r_i, t_i) in the original response and its aligned triple $(h_{i,j}, r_{i,j}, t_{i,j})_{\text{a}}$ in j-th sampled response, $\mathbf{e}_i$ and $\mathbf{e}^{\text{a}}_{i,j}$ are the node embeddings of the tail entity t_i and $t_{i,j}$ after RGCN processing. We compute the similarity between $\mathbf{e}_i$ and $\mathbf{e}^{\text{a}}_{i,j}$. If their similarity $S(\mathbf{e}_i, \mathbf{e}^{\text{a}}_{i,j})$ exceeds the threshold θ_t, we increase the consistency score $c_{i,j}$ of (h_i, r_i, t_i) by 1. This indicates that there is a triple consistent with the triple (h_i, r_i, t_i) in j-th sampled response.

Conversely, we use $\mathbf{e}^{\text{m}}_{t_{i,j}}$ to denote the node embedding of the tail entity in the unaligned triple $(h_{i,j}, r_{i,j}, t_{i,j})_{\text{m}}$ in the j-th sampled response. If the similarity between $\mathbf{e}^{\text{m}}_{t_{i,j}}$ and $\mathbf{e}_i$ exceeds the threshold θ_t, we update the consistency score $c_{i,j}$ of (h_i, r_i, t_i) by subtracting 1. It indicates that the triple may have a risk of hallucination. Note we do not directly label the triple as a hallucination, as two triples with the same head and tail entities but different relations can both be factually correct. Moreover, such cases are rare (1.9% in two datasets for hallucination detection), as the knowledge triples we compare for consistency come from repeated responses to the same query, which are likely to focus on describing the same subject matter. In Sec. 3.4, we also provide a detailed detection for each triple to ensure the accuracy of the results.

$$c_{i,j} = \begin{cases} c_{i,j} + 1 & \text{if S}(\mathbf{e}_{t_i}, \mathbf{e}^{\text{a}}_{t_{i,j}}) > \theta_t \\ c_{i,j} - 1 & \text{if S}(\mathbf{e}_{t_i}, \mathbf{e}^{\text{m}}_{t_{i,j}}) > \theta_t \\ c_{i,j} & \text{otherwise} \end{cases} \quad (2)$$

To obtain the final consistency score for each triple in the response, we sum its comparison results with each sampled response as $C_i = \sum_{j=1}^{n} (c_{i,j})$

During the message passing and aggregation process with RGCN on a graph, each node integrates features from its neighboring nodes. This allows triples containing the node to aggregate contextual information from surrounding triples, Considering the dependencies between the triple to be verified and the surrounding triples.

3.4 Reverse Verification via Triple Reconstruction

To address the omission issue mentioned in Sec. 3.3, we propose a LLM-based reverse verification method (**RVF**), which contains three reconstruction tasks that check whether LLM can reconstruct the knowledge triples' head entity, relation, and tail entity, respectively. Traditional reverse strategies prompt the LLMs to reconstruct questions to verify each knowledge fact from generated responses. The reconstructed question may have multiple correct answers, which leads to a low probability of answering the facts that we aim to verify. It increases the chance of misjudging these facts. To address this, we add constraints to the reconstructed questions to reduce the space of correct answers and increase the probability of answering the triples we want to verify. The three tasks are as follows:

- **Head Entity with Question Answering Task (HEQA).** We prompt LLMs to reconstruct a question for each triple, with the head entity as the expected answer, and then obtain the model's responses. We check if these responses are consistent with the head entity. Specifically, to reduce the space of correct answers for reconstructed questions, we first follow the method from (Manakul, Liusie, and Gales 2023) to initially verify the triples in the original responses. Then, we filter out a set of triples f_t with high factual accuracy. For each triple (h_i, r_i, t_i) in the original response, we add f_t (excluding (h_i, r_i, t_i) if it exists in f_t) as constraints in the questions during the LLM reconstruction process. The model's responses to the question must satisfy these constraints. We repeatedly prompt the LLM to generate answers A to the question. The total number of A denoted as N_A. We count the times that the model responses match the head entity h_i (denoted as N_h) and calculate the fact score S_h as the ratio of N_h to N_A, where $S_h = \frac{N_h}{N_A}$.
- **Relation Regeneration Task (RR).** We mask the relation in the triple with a special token and prompt the model to predict multiple times. Then we check whether the model's predictions are identical to the relation for measuring the consistency. It can reduce the space of correct answers because the relationship between two entities is limited. Specifically, for each triple (h_i, r_i, t_i), we mask r_i with a special token and prompt the LLM for multiple times to predict the original r_i given h_i and t_i. We define the fact score S_r as the proportion of the predicted relations that match the original relation r_i, where $S_r = \frac{N_c}{N_p}$. Here, N_c is the number of matched predictions, and N_p is the total number of predictions.
- **Fact Triple Selection based on Tail Entity Task (FTSTE).** Models often generate long texts centered around a few key entities, which typically serve as the head entities in extracted triples. The limited number of head entities allows us to use surrounding context related to the head entity as constraints to effectively narrow down the space of correct answers for reconstructed questions. However, tail entities in long-text responses are more diverse, so we cannot directly use surrounding contexts as constraints in reconstructed questions. Instead, we use a direct approach by providing a list of options to limit the space of correct answers. We prompt the model to select the factual triple from it. Then, we compare if the model's selections are consistent with the original triple. It reduces the space of correct answers by providing a limited set of choices. Specifically, for each triple (h_i, r_i, t_i), we replace t_i with other entities of the same type to generate multiple similar triplets; and then, we prompt the LLM to choose the factual one. We define the fact score S_t as the proportion of times (h_i, r_i, t_i) is selected, where $S_t = \frac{N_t}{N_s}$. N_t is the number of times (h_i, r_i, t_i) is selected, and N_s is the total number of selections. See the prompt templates used in the above three tasks in App.B.

Finally, we sum up the fact scores from these three tasks and the consistency score mentioned in Sec. 3.3 with different weights to make a judgment about each triple in the original response, as shown in Eq. 3

$$F(h_i, r_i, t_i) = w_1 \cdot S_h + w_2 \cdot S_r + w_3 \cdot S_t + w_4 \cdot C_i \quad (3)$$

In our proposed reverse detection method, the three tasks use different strategies to reduce the space of correct answers in the reconstructed questions. It avoids the issue in traditional reverse detection techniques where the reconstructed questions may have multiple correct answers making it difficult to detect specific facts, improving the accuracy of detecting each triple in the original response.

4 Experiments

4.1 Experimental Setting

Datasets We utilize three datasets: (1) **PHD** (Yang, Sun, and Wan 2023): The dataset consists of 300 samples. Each sample is a Wikipedia article about an entity generated by ChatGPT (gpt-3.5-turbo) and annotated by human annotators. (2) **WikiBio** (Manakul, Liusie, and Gales 2023): The dataset consists of 238 passages generated by GPT3 and annotated at the sentence level. Although it lacks passage-level labels, we follow (Yang, Sun, and Wan 2023) to aggregate sentence labels to derive pseudo-labels at the passage level. (3) **sub-WikiBio**: There are only 12 fact samples in the WikiBio dataset. The sample distribution is too imbalanced. Therefore, we extract all 12 fact samples and 48 randomly selected hallucination samples to create a subset. In our experiment, we refer to it as the WikiBio subset.

Implemention Details We use the recorded responses for each sample as original responses and generate 10 additional sampled responses using ChatGPT. we set the generation temperature to 1.0 to ensure the randomness of sampled responses. We use GPT-4 (gpt-4-1106-preview) to extract triple knowledge from responses and reconstruct questions in reverse verification. At this point, we set the temperature to 0.0 to maximize the reproducibility of the result.

Baselines We compare our method against six baselines: (1) **Reverse Validation via QG (RVQG)** (Yang, Sun, and Wan 2023) is a method that uses LLMs to reconstruct a question about the text to be verified. It compares if the model's response to the reconstructed question is consistent with the text (2) **Semantic Entropy (SE)** (Farquhar et al. 2024) breaks down the entire response into factual claims and prompts LLMs to reconstruct questions about it. For each claim, they repeatedly ask LLM reconstructed questions. And then cluster the claim and the model's responses. They measure the entropy of the cluster containing the claim to assess its validity. (3) **SelfCheckGPT via BERTScore (SelfCk-BS)** (Manakul, Liusie, and Gales 2023) is a variant of SelfCheckGPT, using BERTScore to measure consistency between original response and sampled responses. (4) **SelfCheckGPT via NLI (SelfCk-NLI)** (Manakul, Liusie, and Gales 2023) is another variant of SelfCheckGPT that uses an NLI model to measure consistency between the original response and the sampled responses. (5) **Self-contradiction (SC)** (Mündler et al. 2024) is a prompting-based framework designed to effectively detect self-contradictory hallucinations. (6) **Focus** (Zhang et al. 2023b) is a white-box hallucination detection method that works by focusing on the properties of key tokens in the response. However, SE, SelfCk-BS, SelfCk-NLI, and Focus all return the likelihood scores of a sample being a hallucination, rather than labels indicating fact or hallucination. To align these methods with our task, we set thresholds for these baselines on different datasets using the same approach as for our method. If a sample score exceeds the threshold, we classify it as a hallucination. Details are in App.C.

Evaluation Metrics. We evaluate how well the method detects hallucinatory responses using metrics: (1) **F1** is the harmonic mean of precision and recall, providing a comprehensive evaluation of the classification performance of the method; (2) **Accuracy** is the proportion of correctly classified samples out of the total number of samples.

Methods	PHD		WikiBio		sub-WikiBio	
	F1	Acc	F1	Acc	F1	Acc
RVQG	52.3	65.3	85.7	79.2	88.2	81.7
SE	35.6	62.7	66.7	52.5	87.9	82.3
SelfCk-NLI	23.7	42.0	60.2	43.3	44.7	30.0
SelfCk-BS	40.5	55.0	71.0	57.1	88.8	83.3
SC	30.9	65.7	75.8	62.2	83.7	76.7
Focus	46.7	62.0	75.7	61.3	83.3	76.7
GCA	**55.4**	**68.3**	**90.7**	**83.2**	**90.5**	**85.0**

Table 1: Evaluation results of all methods.

4.2 Overall Performance

We analyze the effectiveness of our method by comparing it with six baselines, results shown in Tab.1. Our method outperforms baselines on all metric values. SelfCk-NLI uses an NLI model to assess if any sentence in the original response contradicts sampled responses and performs the worst on all metrics. SelfCk-NLI does not perform as well as SelfCk-BS, suggesting that NLI models have limited ability to compare consistency between texts. It is even less effective than assessing via simple embedding similarity measures. Reverse validation methods (RVQG and SE) perform worse than our method on all metrics. We attribute this to using a graph-based consistency comparison method (Sec. 3.3), which considers dependencies between triples during comparison. Notably, our method outperforms Focus, a white-box hallucination detection method that uses internal model information, further demonstrating its outstanding efficacy.

4.3 Ablation study

We conduct an ablation study to verify the importance of each component as shown in Tab 2. $-$ *RVF* means abandoning the reverse validation from our full model. The performance drop across most datasets indicates that RVF effectively addresses the omission issues in GCCC to improve the overall effectiveness. However, the performance does not drop on the WikiBio. The reason is that WikiBio contains many hallucination samples (95%), causing our method, baselines, and their variants to show a bias toward predicting hallucinations in this dataset. In these abnormal conditions, the RVF module does not perform effectively, as its advantage lies in correctly identifying hallucination samples. With a more balanced sample distribution in the dataset (sub-WikiBio), our full model performs better than GCCC as expected. $-$ *GCCC* removes GCCC from the full

Variants	PHD		WikiBio		sub-WikiBio	
	F1	Acc	F1	Acc	F1	Acc
− RVF	38.1	40.3	**90.7**	**83.2**	88.7	80.0
− GCCC	54.0	67.7	87.1	78.2	87.6	81.7
− RR	52.1	65.6	87.0	77.7	90.0	83.3
− FTSTE	52.1	66.3	86.8	77.7	89.1	83.3
− HEQA	36.4	54.6	84.5	73.9	85.1	75.0
− Relations	53.7	66.7	86.8	77.7	88.6	80.0
− Graph	52.8	66.7	83.7	73.1	87.6	81.7
GCA	**55.4**	**68.3**	**90.7**	**83.2**	**90.5**	**85.0**

Table 2: Ablation studies on model components. −RVF and −GCCC respectively means detecting without RVF and GCCC. − RR, − FTSTE and − HEQA respectively indicate removing the RR task, FTSTE task, and HEQA task from the full model. − Relations means detecting without relations in triples. − Graph means detecting without graph network model.

model, performing worse than GCA. It indicates that GCA utilizes GCCC to consider the dependencies between triples in the consistency comparison process, improving the accuracy of results.− *RR*, − *FTSTE* and − *HEQA* respectively represent removing the RR task, FTSTE task, and HEQA task mentioned in Sec.3.4 from our full model. − HEQA shows the worst performance, indicating that the HEQA task is the most effective reverse detection strategy. − *Relations* means not utilizing the relations in the triples during the consistency comparison process. It replaces the RGCN used in GCCC with GCN, and the results show a decline. It suggests that relation information is useful and RGCN effectively integrates it for each triple. − *Graph* means not using graph network, performing worse than GCA. It indicates that information integrated by RGCN is beneficial for detection.

4.4 Analysis on Contextual Integration in GCCC

To verify that our graph-based method effectively aggregates node information, we design an experiment to compare two scenarios: **(S1)** using RGCN for message passing and aggregation on the graph; **(S2)** without RGCN, examining the similarity between nodes and their surrounding neighbors. Specifically, we conduct two experiments as follows:

t-SNE Visualization of Node Representation Distribution. The first experiment uses t-SNE dimensionality reduction to project the node representations from both scenarios into a two-dimensional space to observe the distribution. Fig.2 shows that in both the PHD and the WikiBio, the node distribution in the **(S1)** (red nodes) is more compact compared to the **(S2)** (blue nodes). This indicates that after using RGCN, the node representations become more similar to those of their neighbors. RGCN effectively integrates the features of neighboring nodes into each node's representation, blending information for every node.

Quantitative Analysis of Node Representation Similarity. We perform a quantitative analysis by obtaining the cosine similarity of node representations under both **(S1)** and **(S2)**. Tab.3 shows that the representations' similarity between two nodes is higher after processing with RGCN compared to without RGCN. This also indicates that our method integrates contextual information for each node by using RGCN.

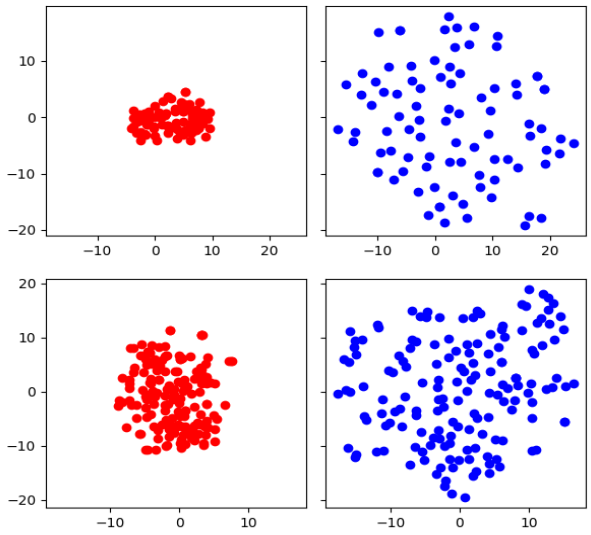

Figure 2: Node distribution comparisons with (red points) and without (blue points) RGCN on the PHD (top) and WikiBio (bottom).

Methods	PHD		WikiBio	
	Min	Avg	Min	Avg
GCCC	0.277	0.486	0.401	0.557
− RGCN	0.189	0.305	0.167	0.247

Table 3: Similarity of representations between nodes. GCCC means our full graph-based module GCCC. − RGCN indicates not using RGCN in the GCCC module. App. D shows the full version of the results.

4.5 Triple Dependencies Error Detection

We design an experiment to verify that our method can also detect errors in triple dependencies. Specifically, we create a new dataset, TripleCom, by selecting samples with errors in triple dependencies extracted from the PHD and WikiBio datasets. The proportion of such error is approximately 10.5% in these datasets. Then we test GCA and six baselines on this dataset, with implementation details matching those in Sec.4.1.2. As shown in Tab. 4, our method GCA achieves the best performance on all metrics in the TripleCom dataset, demonstrating its effectiveness in detecting errors in the dependencies between multiple triples.

5 Conclusion

In this paper, we propose a graph-based context-aware hallucination detection method on long-text generation, where our method follows a zero-resource setting and uses only black-box LLMs. Our method extracts knowledge triples

Methods	F1	Acc
RVQG	81.1	68.2
SE	56.3	39.1
SelfCk-BS	66.7	50.0
SelfCk-NLI	66.7	50.0
SC	70.6	54.5
Focus	75.7	60.1
GCA	**92.7**	**86.3**

Table 4: Results of methods on the TripleCom dataset.

from output responses for better alignment. We then construct a graph to carry contextual information so that considers dependencies between knowledge triples. It indeed addresses the issue of ignoring the contextual information in existing methods that only focus on individual facts. We construct three reconstruction tasks for reverse verification to verify the knowledge triples. Experiments show that our method outperforms all baselines, including the white-box method with access to internal model information, excelling in hallucination detection.

Acknowledgements

This work is supported by the following fundings: National Natural Science Foundation of China under Grant No. 62376284 and No. 62306330, Young Elite Scientist Sponsorship Program by CAST (2023QNRC001) under Grant No. YESS20230367.

References

Allen, B.; Polat, F.; and Groth, P. 2024. SHROOM-INDElab at SemEval-2024 Task 6: Zero- and Few-Shot LLM-Based Classification for Hallucination Detection. In Ojha, A. K.; Doğruöz, A. S.; Tayyar Madabushi, H.; Da San Martino, G.; Rosenthal, S.; and Rosá, A., eds., *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, 839–844. Association for Computational Linguistics.

Azaria, A.; and Mitchell, T. 2023. The Internal State of an LLM Knows When It's Lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 967–976.

Bayat, F. F.; Qian, K.; Han, B.; Sang, Y.; Belyy, A.; Khorshidi, S.; Wu, F.; Ilyas, I.; and Li, Y. 2023. FLEEK: Factual Error Detection and Correction with Evidence Retrieved from External Knowledge. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 124–130.

Cao, Z.; Yang, Y.; and Zhao, H. 2024. AutoHall: Automated Hallucination Dataset Generation for Large Language Models. arXiv:2310.00259.

Chen, C.; Liu, K.; Chen, Z.; Gu, Y.; Wu, Y.; Tao, M.; Fu, Z.; and Ye, J. 2024. INSIDE: LLMs' Internal States Retain the Power of Hallucination Detection. arXiv:2402.03744.

Cohen, R.; Hamri, M.; Geva, M.; and Globerson, A. 2023. LM vs LM: Detecting Factual Errors via Cross Examination. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 12621–12640.

Ding, H.; Pang, L.; Wei, Z.; Shen, H.; and Cheng, X. 2024. Retrieve Only When It Needs: Adaptive Retrieval Augmentation for Hallucination Mitigation in Large Language Models. arXiv:2402.10612.

Fadeeva, E.; Rubashevskii, A.; Shelmanov, A.; Petrakov, S.; Li, H.; Mubarak, H.; Tsymbalov, E.; Kuzmin, G.; Panchenko, A.; Baldwin, T.; Nakov, P.; and Panov, M. 2024. Fact-Checking the Output of Large Language Models via Token-Level Uncertainty Quantification. arXiv:2403.04696.

Fang, M.; Ji, S.; Zuo, J.; Huang, H.; Xia, Y.; Zhu, J.; Cheng, X.; Yang, X.; Liu, W.; Wang, G.; Dong, Z.; and Zhao, Z. 2024. ACE: A Generative Cross-Modal Retrieval Framework with Coarse-To-Fine Semantic Modeling. arXiv:2406.17507.

Farquhar, S.; Kossen, J.; Kuhn, L.; and Gal, Y. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017): 625–630.

Friel, R.; and Sanyal, A. 2023. Chainpoll: A high efficacy method for LLM hallucination detection. arXiv:2310.18344.

Hegselmann, S.; Shen, S. Z.; Gierse, F.; Agrawal, M.; Sontag, D.; and Jiang, X. 2024. A Data-Centric Approach To Generate Faithful and High Quality Patient Summaries with Large Language Models. arXiv:2402.15422.

Hu, X.; Ru, D.; Qiu, L.; Guo, Q.; Zhang, T.; Xu, Y.; Luo, Y.; Liu, P.; Zhang, Y.; and Zhang, Z. 2024. RefChecker: Reference-based Fine-grained Hallucination Checker and Benchmark for Large Language Models. arXiv:2405.14486.

Huang, Q.; Tao, M.; Zhang, C.; An, Z.; Jiang, C.; Chen, Z.; Wu, Z.; and Feng, Y. 2023. Lawyer LLaMA Technical Report. arXiv:2305.15062.

Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y. J.; Madotto, A.; and Fung, P. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12): 1–38.

Jiang, C.; Qi, B.; Hong, X.; Fu, D.; Cheng, Y.; Meng, F.; Yu, M.; Zhou, B.; and Zhou, J. 2024. On Large Language Models' Hallucination with Regard to Known Facts. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 1041–1053.

Jiang, Z.; Araki, J.; Ding, H.; and Neubig, G. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9: 962–977.

Kang, H.; Ni, J.; and Yao, H. 2024. Ever: Mitigating Hallucination in Large Language Models through Real-Time Verification and Rectification. arXiv:2311.09114.

Li, M.; Wang, W.; Feng, F.; Zhu, F.; Wang, Q.; and Chua, T.-S. 2024a. Think Twice Before Trusting: Self-Detection for Large Language Models through Comprehensive Answer Reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 11858–11875.

Li, N.; Li, Y.; Liu, Y.; Shi, L.; Wang, K.; and Wang, H. 2024b. Drowzee: Metamorphic Testing for Fact-Conflicting Hallucination Detection in Large Language Models. arXiv:2405.00648.

Liu, Y.; Iter, D.; Xu, Y.; Wang, S.; Xu, R.; and Zhu, C. 2023. G-Eval: NLG Evaluation using Gpt-4 with Better Human Alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2511–2522.

Liu, Z.; Kou, B.; Li, P.; Yan, M.; Zhang, J.; Huang, F.; and Liu, Y. 2024. Enabling Weak LLMs to Judge Response Reliability via Meta Ranking. arXiv:2402.12146.

Lu, M.; Huang, Z.; Li, B.; Zhao, Y.; Qin, Z.; and Li, D. 2022. Sifter: A framework for robust rumor detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30: 429–442.

Lu, M.; Huang, Z.; Tian, Z.; Zhao, Y.; Fei, X.; and Li, D. 2023a. Meta-tsallis-entropy minimization: a new self-training approach for domain adaptation on text classification. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 5159–5169.

Lu, M.; Huang, Z.; Zhao, Y.; Tian, Z.; Liu, Y.; and Li, D. 2023b. DaMSTF: Domain Adversarial Learning Enhanced Meta Self-Training for Domain Adaptation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1650–1668.

Luo, J.; Xiao, C.; and Ma, F. 2023. Zero-Resource Hallucination Prevention for Large Language Models. arXiv:2309.02654.

Ma, J.; Dai, D.; Sha, L.; and Sui, Z. 2024. Large Language Models Are Unconscious of Unreasonability in Math Problems. arXiv:2403.19346.

Manakul, P.; Liusie, A.; and Gales, M. 2023. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 9004–9017.

Mündler, N.; He, J.; Jenko, S.; and Vechev, M. 2024. Self-contradictory Hallucinations of Large Language Models: Evaluation, Detection and Mitigation. arXiv:2305.15852.

Nahar, M.; Seo, H.; Lee, E.-J.; Xiong, A.; and Lee, D. 2024. Fakes of Varying Shades: How Warning Affects Human Perception and Engagement Regarding LLM Hallucinations. arXiv:2404.03745.

Rawte, V.; Tonmoy, S. M. T. I.; Rajbangshi, K.; Nag, S.; Chadha, A.; Sheth, A. P.; and Das, A. 2024. FACTOID: FACtual enTailment fOr hallucInation Detection. arXiv:2403.19113.

Roychowdhury, S.; Krema, M.; Mahammad, A.; Moore, B.; Mukherjee, A.; and Prakashchandra, P. 2024. ERATTA: Extreme RAG for Table To Answers with Large Language Models. arXiv:2405.03963.

Sadat, M.; Zhou, Z.; Lange, L.; Araki, J.; Gundroo, A.; Wang, B.; Menon, R.; Parvez, M.; and Feng, Z. 2023. DelucionQA: Detecting Hallucinations in Domain-specific Question Answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 822–835.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, 593–607. Springer.

Snyder, B.; Moisescu, M.; and Zafar, M. B. 2024. On early detection of hallucinations in factual question answering. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2721–2732.

Su, X.; Huang, Z.; Su, Y.; Trisedya, B. D.; Dou, Y.; and Zhao, Y. 2024. Hierarchical Shared Encoder with Task-specific Transformer Layer Selection for Emotion-Cause Pair Extraction. *IEEE Transactions on Affective Computing*.

Tian, Z.; Wang, Y.; Song, Y.; Zhang, C.; Lee, D.; Zhao, Y.; Li, D.; and Zhang, N. L. 2022. Empathetic and Emotionally Positive Conversation Systems with an Emotion-specific Query-Response Memory. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 6364–6376.

Varshney, N.; Yao, W.; Zhang, H.; Chen, J.; and Yu, D. 2023. A Stitch in Time Saves Nine: Detecting and Mitigating Hallucinations of LLMs by Validating Low-Confidence Generation. arXiv:2307.03987.

Verma, S.; Tran, K.; Ali, Y.; and Min, G. 2023. Reducing LLM Hallucinations using Epistemic Neural Networks. arXiv:2312.15576.

Wan, H.; Feng, S.; Tan, Z.; Wang, H.; Tsvetkov, Y.; and Luo, M. 2024. DELL: Generating Reactions and Explanations for LLM-Based Misinformation Detection. arXiv:2402.10426.

Wang, H.; Tian, Z.; Song, X.; Zhang, Y.; Pan, Y.; Tu, H.; Huang, M.; and Zhou, B. 2024a. Intent-Aware and Hate-Mitigating Counterspeech Generation via Dual-Discriminator Guided LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 9131–9142.

Wang, W.; Wei, F.; Dong, L.; Bao, H.; Yang, N.; and Zhou, M. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33: 5776–5788.

Wang, X.; Pan, J.; Ding, L.; and Biemann, C. 2024b. Mitigating Hallucinations in Large Vision-Language Models with Instruction Contrastive Decoding. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Wang, Y.; Hu, M.; Huang, Z.; Li, D.; Luo, W.; Yang, D.; and Lu, X. 2023a. A canonicalization-enhanced known fact-aware framework for open knowledge graph link prediction. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2332–2342.

Wang, Y.; Huang, Z.; Hu, M.; Li, D.; Lu, X.; Luo, W.; and Yang, D. 2023b. Structure Enhanced Path Reasoning for Knowledge Graph Completion. *International Journal of Intelligent Systems*, 2023(1): 3022539.

Wen, Z.; Tian, Z.; Huang, Z.; Yang, Y.; Jian, Z.; Wang, C.; and Li, D. 2023a. GRACE: gradient-guided controllable retrieval for augmenting attribute-based text generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, 8377–8398.

Wen, Z.; Tian, Z.; Jian, Z.; Huang, Z.; Ke, P.; Gao, Y.; Huang, M.; and Li, D. 2024. Perception of Knowledge Boundary for Large Language Models through Semi-open-ended Question Answering. arXiv:2405.14383.

Wen, Z.; Tian, Z.; Wu, W.; Yang, Y.; Shi, Y.; Huang, Z.; and Li, D. 2023b. GROVE: A Retrieval-augmented Complex Story Generation Framework with A Forest of Evidence. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 3980–3998.

Xue, T.; Wang, Z.; Wang, Z.; Han, C.; Yu, P.; and Ji, H. 2023. RCOT: Detecting and Rectifying Factual Inconsistency in Reasoning by Reversing Chain-of-Thought. arXiv:2305.11499.

Yang, S.; Sun, R.; and Wan, X. 2023. A New Benchmark and Reverse Validation Method for Passage-level Hallucination Detection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 3898–3908.

Yehuda, Y.; Malkiel, I.; Barkan, O.; Weill, J.; Ronen, R.; and Koenigstein, N. 2024. InterrogateLLM: Zero-Resource Hallucination Detection in LLM-Generated Answers. arXiv:2403.02889.

Yuksekgonul, M.; Chandrasekaran, V.; Jones, E.; Gunasekar, S.; Naik, R.; Palangi, H.; Kamar, E.; and Nushi, B. 2024. Attention Satisfies: A Constraint-Satisfaction Lens on Factual Errors of Language Models. arXiv:2309.15098.

Zhang, J.; Li, Z.; Das, K.; Malin, B.; and Kumar, S. 2023a. SAC3: Reliable Hallucination Detection in Black-Box Language Models via Semantic-aware Cross-check Consistency. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 15445–15458.

Zhang, T.; Qiu, L.; Guo, Q.; Deng, C.; Zhang, Y.; Zhang, Z.; Zhou, C.; Wang, X.; and Fu, L. 2023b. Enhancing Uncertainty-Based Hallucination Detection with Stronger Focus. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 915–932.

Zhang, Y.; Du, L.; Cao, D.; Fu, Q.; and Liu, Y. 2024. An Examination on the Effectiveness of Divide-and-Conquer Prompting in Large Language Models. arXiv:2402.05359.

Zhu, D.; Chen, D.; Li, Q.; Chen, Z.; Ma, L.; Grossklags, J.; and Fritz, M. 2024. PoLLMgraph: Unraveling Hallucinations in Large Language Models via State Transition Dynamics. In *Findings of the Association for Computational Linguistics: NAACL 2024*, 4737–4751.