SiGHT: A Self-Supervised Graph-based Hallucination DeTection Framework for Domain-Specific LLMs

Zi-Ying Chen
National Yang Ming Chiao Tung University
Department of Computer Science
Hsinchu, Taiwan
ziyingchen.cs12@nycu.edu.tw

Meng-Fen Chiang
National Yang Ming Chiao Tung University
Department of Electrical Engineering
Hsinchu, Taiwan
meng.chiang@nycu.edu.tw

Wen-Chih Peng
National Yang Ming Chiao Tung University
Department of Computer Science
Hsinchu, Taiwan
wcpengcs@nycu.edu.tw

Abstract

Factual reliability in domain-specific Large Language Models (LLMs) is paramount in high-stakes applications where incorrect outputs carry significant risks. Current detection methodologies often rely on expensive retrieval validation or labor-intensive manual annotation, creating substantial barriers to scalable deployment. To bridge the gap, we propose SiGHT, a self-supervised graph framework designed for efficient hallucination detection in specialized contexts. SiGHT introduces an automated training pipeline that leverages prompt strategies to synthesize plausible hallucinated content from structured knowledge, effectively eliminating the need for human labeling. By mapping texts to high-resolution word-level relational graphs, the framework employs a Graph Attention Network (GAT) to model fine-grained semantic dependencies and identify structural inconsistencies. Empirical evaluations on the MSMARCO-QnA and RAGTruth-QA benchmarks demonstrate that SiGHT achieves a 46.94% relative F1 gain over prior graph baselines. Notably, SiGHT remains competitive with state of the art detectors while utilizing only 0.03M parameters and incurring a minimal inference latency of 0.342 seconds per instance. Dominating the accuracy–efficiency frontier, SiGHT delivers a robust and scalable architecture for real-time hallucination monitoring in high-stakes specialized pipelines.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

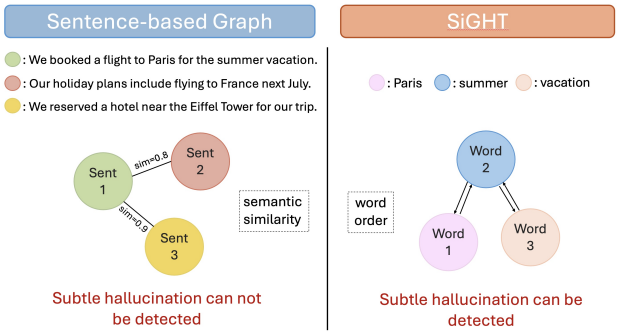

Figure 1: **Representation granularity comparison.** While sentence-level graphs aggregate semantics at a coarse resolution, SiGHT employs token-level graphs to resolve subtle hallucinations via high-resolution relational dependencies.

1 INTRODUCTION

Large language models (LLMs) have rapidly reshaped the AI landscape (Hagos et al., 2024) and are now widely adopted as general-purpose tools across NLP tasks (Zhao et al., 2023). Yet their tendency to produce hallucinations, i.e., fluent but factually incorrect outputs (Huang et al., 2025; Tonmoy et al., 2024; Sriramanan et al., 2024), undermines reliability, especially in domain-specific, high-stakes settings such as medicine and law (Huang et al., 2025; Wang et al., 2023). Mitigation via Retrieval-Augmented Generation (RAG) (Chen et al., 2024a) or fine-tuning (Parthasarathy et al., 2024) reduces but does not eliminate errors (Chen et al., 2024a; Gekhman et al., 2024), making robust detection a first-class requirement for workflows where latency, cost, and accountability matter.

Existing hallucination detectors encompass a range of approaches, including internal-state classifiers that train on hidden activations (Azaria and Mitchell,

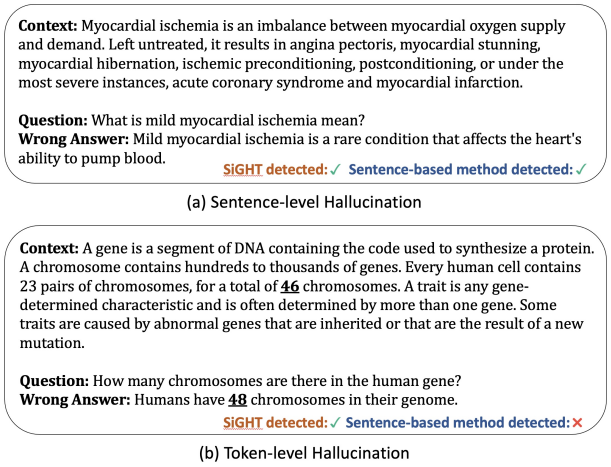

Figure 2: **Hallucination detection across granularities.** Examples of (a) sentence-level and (b) token-level hallucinations. SiGHT effectively identifies factual inconsistencies, such as numerical entity replacements, that remain latent to sentence-based detectors due to high surface-level similarity.

2023), as well as strong cross-encoder baselines such as HHEM[1][2]. However, these approaches depend on large, manually annotated corpora, creating a **data scarcity challenge**. Annotation-free alternatives lean on LLMs or retrieval at inference, e.g., SelfCheckGPT (Manakul et al., 2023), Chain-of-Verification (CoVe) (Dhuliawala et al., 2023), and retrieval-enhanced verification (Huo et al., 2023), thus incurring prompt/LLM cost, retrieval fragility, and latency, which pose a **scalability challenge**. Graph-based detectors have gained traction by exploiting structure—via curated knowledge graphs (Furumai et al., 2023; Sansford et al., 2024) or sentence-level, annotation-free graphs (Nonkes et al., 2024)—but either require labor-intensive KG construction or operate at too coarse a resolution to capture token-level inconsistencies, leading to a **representation granularity challenge**. We therefore argue that effective deployment in constrained, domain-specific settings demands detectors that are *annotation-free*, *token-granular*, and *inexpensive at inference*.

To overcome these challenges, we propose **SiGHT**, a novel Self-supervised Graph-based Hallucination deTection framework designed for domain-specific contexts. SiGHT comprises two key stages: (1) *hallucination detection model training*, and (2) *dialogue hallucination detection*. In stage one, SiGHT begins with domain textual data, learns word-level representations using Word2Vec (Church, 2017), and generates training samples without manual labels by prompting an LLM to produce plausible but incorrect variants of factual text. Both factual and synthesized texts are encoded as word-level relational graphs that preserve local and long-range dependencies (Figure 1). In contrast to sentence-based graph construction (Nonkes et al., 2024), this fine-grained representation exposes subtle token-level contradictions (see Figure 2). A compact Graph Attention Network (Veličković et al., 2017) then aggregates informative neighborhoods using multi-head attention to learn a detector sensitive to token-scale inconsistencies. In stage two, the trained graph encoder directly classifies LLM responses, avoiding external LLM queries and retrieval at inference, which reduces latency and cost while maintaining accuracy. We evaluate SiGHT on MMSMARCO-QnA (Bajaj et al., 2016) and RAGTruth-QA (Niu et al., 2023). SiGHT outperforms graph-based baselines by 46.94% in F1 on MSMARCO-QnA and achieves 45.90% F1 on RAGTruth-QA with only 0.03M parameters and 0.342 seconds per instance, demonstrating favorable accuracy and efficiency for domain-specific deployment.

Our main contributions are summarized as follows:

- We propose SiGHT, a novel self-supervised pipeline that eliminates the need for manual annotation by synthesizing training negatives from structured domain knowledge.
- We propose a token-granular, word-level graph representation, combined with a compact graph attention detector, that captures fine-grained relational semantics.
- We provide empirical evidence on the MSMARCO-QnA and RAGTruth-QA benchmarks, demonstrating strong detection performance with a minimal parameter footprint and low inference latency.

2 PROBLEM FORMULATION

In this section, we formally define the constituent elements and establish the mathematical basis for graph-based hallucination detection.

Definition 2.1 (**Domain Corpus**). *Let $\mathcal{D}$ be a domain-specific corpus decomposed into paragraphs $\mathcal{P} = \{p_1, \ldots, p_n\}$. We define a processing function $\Phi(\cdot)$ that performs tokenization and POS-based filtering to extract a set of semantically significant tokens $V_p = \Phi(p)$ for each paragraph.*

Definition 2.2 (**Word-level Graph**). *For each paragraph p, we construct a bidirectional word graph $G = (V_p, E_p, \mathbf{X}_p)$. Here, E_p represents edges capturing the sequential order of tokens, and $\mathbf{X}_p \in \mathbb{R}^{|V_p| \times d}$ is the node feature matrix where each row $\mathbf{x}_w$ is a d-dimensional embedding for token $w \in V_p$.*

Problem (**Hallucination Detection**). *We formulate hallucination detection as a binary classification task*

[1] Vectara HHEM v2: a new and improved factual consistency scoring model

[2] Vectara huggingface: hallucination evaluation model

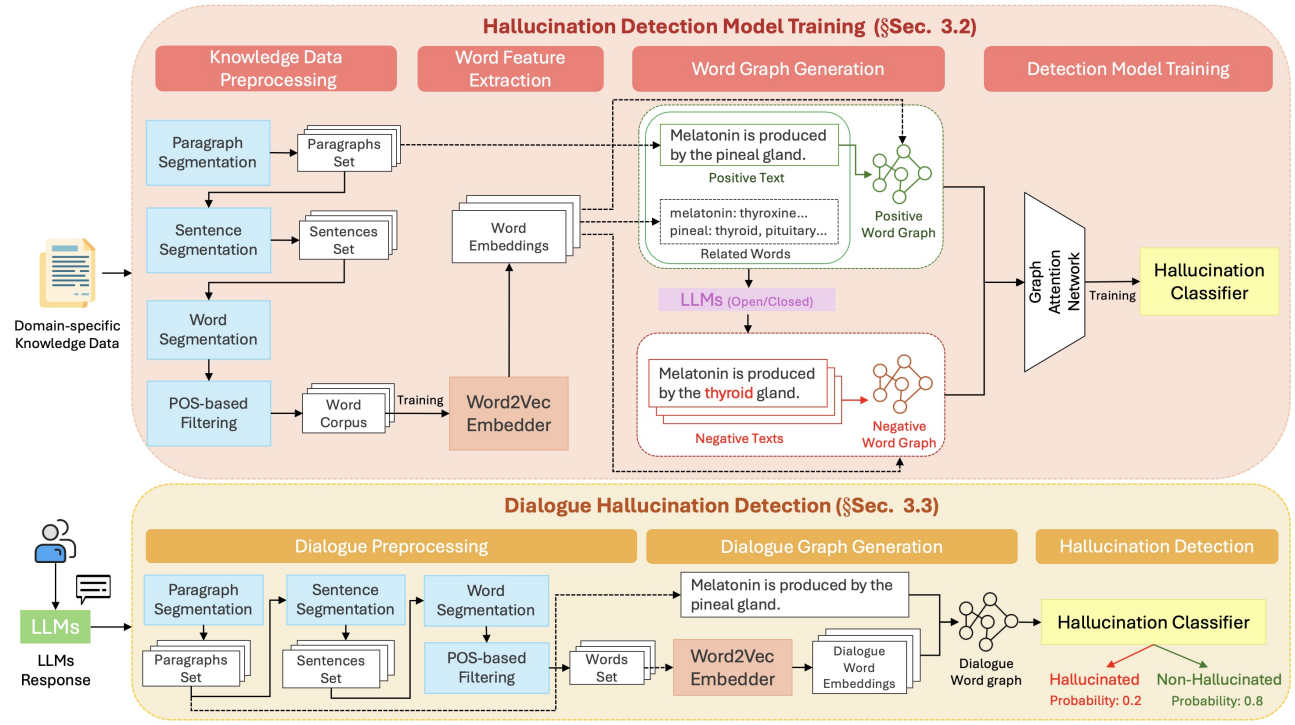

Figure 3: **Overview of SiGHT.** (i) Hallucination Detection Model Training: A hallucination detection model is trained in a self-supervised manner, where domain-specific knowledge is transformed into word-level graphs to capture fine-grained features via a Graph Attention Network (GAT); (ii) Dialogue Hallucination Detection: The trained model classifies LLM-generated dialogue as hallucinated or non-hallucinated based on the learned graph-based representations.

performed over the domain of word graphs. Given a domain corpus $\mathcal{D}$ and its induced graph set $\mathcal{G} = \mathcal{G}^+ \cup \mathcal{G}^-$, where $\mathcal{G}^+$ denotes factual graphs and $\mathcal{G}^-$ denotes hallucinated graphs, the objective is to learn a mapping $\mathcal{F}_\Omega : G \to [0, 1]$ that estimates the hallucination probability $\hat{y}$:

$$\hat{y} = \mathcal{F}_\Omega(G) \approx P(y = 1 \mid G), \tag{1}$$

where Ω signifies the set of trainable parameters. Under this formulation, a ground truth label $y = 1$ is assigned if $G \in \mathcal{G}^-$ and $y = 0$ if $G \in \mathcal{G}^+$. The detector must generalize across these structural representations to identify subtle factual deviations that lack clear linguistic markers.

3 METHODOLOGY

We introduce SiGHT, a self-supervised framework for hallucination detection tailored for domain-specific LLMs. The architecture comprises two primary stages: (i) Hallucination Detection Model Training, which leverages automated negative synthesis and graph representation learning, and (ii) Dialogue Hallucination Detection, which enables efficient, single-pass inference for real-time applications.

3.1 Framework Overview

The hallucination-detection model training stage eliminates the need for manual labels by leveraging self-supervised negative synthesis. By generating plausible yet factually incorrect variants from the source corpus, SiGHT constructs factual and counterfactual word-level graph pairs, thereby broadening error coverage while minimizing annotation costs. Each instance is modeled as a relational graph that preserves local token interactions. A graph attention encoder is trained to identify local inconsistency cues, exposing token- and span-level scores at inference to recover fine-grained signals that coarse sentence-level features typically overlook. Next, the dialogue detection stage resolves heavy inference and restricted access by applying a compact graph attention classifier to a single response graph in a single pass, with no retrieval, no auxiliary LLM calls, and no internal hooks, enabling low-latency external deployment. Figure 3 summarizes the framework.

3.2 Hallucination Detection Model Training

Knowledge Data Pre-processing. Domain knowledge is often unstructured and noisy, requiring careful

pre-processing to support a subsequent graph-based hallucination classification. Additionally, to capture fine-grained relational semantic modeling from domain knowledge documents, we design a systematic pre-processing pipeline to normalize and clean the data for constructing structured graphs. The pipeline comprises four sequential operators applied in order: paragraph segmentation ParaSeg(·), sentence segmentation SentSeg(·), word tokenization WordSeg(·), and part of speech filtering POSFilter(·). Specifically, the paragraphs are first segmented to preserve the topical coherence and reduce semantic interference. Each paragraph is then split into sentences, followed by word-level tokenization with standard cleaning procedures (e.g., removing redundant tokens, URLs, and whitespace). Finally, POS-based filtering retains only semantically meaningful tokens (e.g., nouns, verbs, adjectives, numerals), while excluding less meaningful elements (e.g., prepositions, particles). The formal pipeline is:

$$V_p = \text{POSFilter}\Big(\text{WordSeg}\big(\text{SentSeg}(\text{ParaSeg}(D))\big)\Big),$$

where V_p represents the filtered token set for paragraph p as defined in Section 2, serving as the node set for subsequent word-level graphs.

Word Feature Extraction. To project tokens into a continuous semantic space that captures syntactic regularities, we train a Word2Vec (Church, 2017) encoder `W2V` on the pre-processed word set. Word2Vec leverages distributional co-occurrence patterns to learn vectors that reflect syntactic and semantic similarity (Pouly et al., 2024). In contrast to contextualized encoders that generate input-dependent representations, Word2Vec yields context-independent lexical embeddings that remain stable across different occurrences of the same token. This stability is critical for our graph construction, where each word functions as a persistent node type shared across multiple graphs. Inconsistent representations across varied contexts can undermine the structural coherence of the learned graph patterns. Previous research indicates that contextualized embeddings from transformer-based models exhibit substantial variance across contexts, making them less suitable for static lexical graphs (Ethayarajh, 2019). Furthermore, training Word2Vec on domain-specific data provides robust representations for specialized terminology, which is vital for detecting hallucinations. Attaining comparable performance with high-capacity transformer models requires extensive fine-tuning and substantial computational resources (Strubell et al., 2019), an overhead fundamentally at odds with our focus on resource efficiency and architectural simplicity. Formally, for each paragraph p, the encoder `W2V` maps each token $w \in V_p$ to a d-dimensional vector $\mathbf{x}_w = E(w) \in \mathbb{R}^d$. These vectors form the node feature matrix $\mathbf{X}_p \in \mathbb{R}^{|V_p| \times d}$, providing the foundational semantic signal for graph modeling.

Word Graph Generation. To leverage the structured inductive biases of graph representations, we transform domain-knowledge texts into graphs that capture both relational and contextual semantics. Given that each paragraph typically encapsulates a coherent topical unit, we treat each paragraph p as an independent graph G, resulting in multiple graphs per document. In each graph, tokens $w \in V_p$ serve as nodes, with attributes defined by the static embeddings $\mathbf{X}_p$ extracted from the trained Word2Vec model `W2V`. Edges E_p are constructed between consecutive word tokens within a paragraph to model sequential dependencies. To account for rephrasing and the varied word-order often introduced by large language models (LLMs) (Wang et al., 2025), we define the edges as bidirectional. This process yields the factual graph set $\mathcal{G}^+$ derived from the domain corpus, assuming the knowledge texts' content is factually verified by the user. To enable self-supervised learning of hallucination cues, we synthesize hallucinated paragraphs $\widehat{p^-}$ using an LLM, $\texttt{LLM}_{\text{gen}}(\cdot)$, guided by a prompt template $\texttt{PT}(\cdot)$ that takes the original paragraph p, a set of top-N nearest neighbors $\mathcal{N}(w)$ for each token w derived from the trained Word2Vec model `W2V`, and instructions that elicit fluent yet factually incorrect variants (details in Appendices A.2 and A.1). There is no restriction on the choice of LLMs; both open- and closed-source models can be used. The overall generation process of the hallucinated paragraph $\widehat{p^-}$ is described as:

$$\widehat{p^-} = \texttt{LLM}_{\text{gen}}\left(\texttt{PT}\left(p, \{\mathcal{N}(w) \mid w \text{ appears in } p\}\right)\right). \quad (2)$$

We apply the same pre-processing and graph construction to $\widehat{p}^-$ to obtain the negative set $\mathcal{G}^-$. The final dataset for model training is the union of positive and negative graphs: $\mathcal{G} = \mathcal{G}^+ \cup \mathcal{G}^-$. The algorithm for constructing $\mathcal{G}^+$ is presented in Algorithm-1, while that for $\mathcal{G}^-$ is presented in Algorithm-2.

Detection Model Training. We frame hallucination detection as a binary classification task. Specifically, we assign label $y = 0$ to factual graph $G \in \mathcal{G}^+$ and label $y = 1$ to synthesized hallucinated graph $G \in \mathcal{G}^-$. To capture the intricate relational and contextual structures within these graphs, we adopt the Graph Attention Network (GAT) (Veličković et al., 2017) as our core classification model architecture due to its several key advantages: (i) GAT is equipped with adaptive attention mechanism over node neighborhoods, allowing our model to focus more on informative neighbors; (ii) GAT can capture complex and fine-grained structural dependencies in a graph; (iii) GAT can enhance stability and expressiveness of node representations through multi-head attention (Veličković et al., 2017).

Algorithm 1: Positive Graph Construction

Input: Factual paragraph P, Word2Vec encoder `W2V`
Output: Factual graph $G^+ = (V, E)$
Initialize token sequence L;
foreach *sentence* $s \in P$ **do**
  Extract filtered tokens and POS tags from s;
  Append to L;
Initialize node set $V \leftarrow \emptyset$, edge set $E \leftarrow \emptyset$;
Let $L = (w_1, \ldots, w_n)$ be the sequence of extracted tokens;
foreach *token* $w_i \in L$ **do**
  if $w_i \in$ `W2V` **then**
    Assign embedding vector $x_i \leftarrow \texttt{W2V}[w_i]$;
  else
    Assign placeholder vector $x_i \leftarrow$ **constant**;
  Add node (w_i, x_i) into V; **if** $i + 1 < |L|$ **then**
    Add edge (w_i, w_{i+1}) into E;
return $G^+ = (V, E)$

Algorithm 2: Negative Graph Construction

Input: Factual paragraph T, token list L, POS dictionary POS, Word2Vec encoder `W2V`, LLM client `LLM`, minimum exchange count $MEWC$, exchange ratio EWR, top-N similarity threshold MST
Output: Hallucinated graph $G^- = (V', E')$
Compute maximum exchange count $ME \leftarrow |L| \times EWR$;
Set replace count $RC \leftarrow \max(MEWC, \text{random}(MEWC, ME))$;
Randomly select RC indices from L w/o replacement;
Form subset $S \subseteq L$ containing these words;
Initialize candidate dictionary C;
foreach $w \in S$ **do**
  Retrieve POS tag $pos(w)$ from POS;
  Retrieve top-N ($N = MST$) semantically similar words of w from `W2V`;
  Filter candidates by POS match: keep only words w' such that $pos(w') = pos(w)$;
  Store filtered candidates in $C[w]$;
Construct prompt `PT` using T, target tokens S, and candidates C;
Invoke $\texttt{LLM}_{\text{gen}}(\texttt{PT})$, obtain hallucinated text T';
Segment T' into token sequence L';
Initialize node set $V' \leftarrow \emptyset$, $E' \leftarrow \emptyset$;
foreach *token* $w'_i \in L'$ **do**
  if $w'_i \in$ `W2V` **then**
    Assign embedding $x'_i \leftarrow \texttt{W2V}[w'_i]$;
  else
    Assign placeholder vector $x'_i \leftarrow$ **constant**;
  Add node (w'_i, x'_i) into V';
  if $i + 1 < |L'|$ **then**
    Add edge $(w'i, w'i + 1)$ into E';
return $G^- = (V', E')$

Given a graph $G = (V_p, E_p, \mathbf{X}_p)$, our classifier $\mathcal{F}_\Omega$ computes a hallucination probability $\hat{y} \in [0, 1]$ through three stacked GAT layers with multi-head attention, followed by ReLU activations and dropout for regularization. The final node representations are aggregated via global mean pooling and passed through a multilayer perceptron (MLP) for prediction. The overall $\mathcal{F}_\Omega$ model is formulated as:

$$\begin{aligned}\hat{y} &= \mathcal{F}_\Omega(G) \\ &= \text{MLP}\left(\text{GlobalMeanPool}\left(\text{GAT}(\mathbf{X}_p, E_p)\right)\right).\end{aligned} \tag{3}$$

To mitigate class imbalance and emphasize difficult samples, we adopt the Focal Loss (Lin et al., 2017) as the training objective:

$$\begin{aligned}\mathcal{L}_{\text{focal}}(\hat{y}, y) = &-\alpha(1-\hat{y})^\gamma y \log \hat{y} \\ &- (1-\alpha)\hat{y}^\gamma(1-y)\log(1-\hat{y}),\end{aligned} \tag{4}$$

where $y \in \{0,\ 1\}$ is the ground truth label, $\hat{y}$ is the predicted probability, $\alpha \in [0, 1]$ balances positive and negative samples, and $\gamma > 0$ modulates the focus on hard examples. Accordingly, the overall training objective, which minimizes the focal loss across all training graphs, is formulated as follows:

$$\min_{\Omega} \sum_{i=1}^{N} \mathcal{L}_{\text{focal}}\left(\hat{y}_i, y_i\right), \quad \text{where} \quad \hat{y}_i = \mathcal{F}_\Omega\left(G_i; \Omega\right). \tag{5}$$

3.3 Dialogue Hallucination Detection

To support real-time evaluation of LLM-generated responses, SIGHT employs an inference component that maps dialogues into the learned structural semantic space. This stage captures the word-level relational cues essential for identifying factual inconsistencies without requiring external retrieval or reference documents at runtime.

Dialogue Pre-processing. In the inference phase, we apply the same transformation $\Phi(\cdot)$ described in Section 2 to the LLM-generated responses. Each response is segmented into paragraphs, from which we extract the set of semantically significant tokens V_p. Following the procedure in Section 3.2, each token $w \in V_p$ is mapped to its stable embedding $\mathbf{x}_w = E(w)$, forming the feature matrix $\mathbf{X}_p$.

Dialogue Graph Generation. For each paragraph p in the dialogue, we construct a word-level graph $G = (V_p, E_p, \mathbf{X}_p)$, where edges E_p capture sequential word order through bidirectional connections. This consistent graph construction preserves the structural patterns learned during the self-supervised training phase at inference time.

Hallucination Detection. The resulting graphs are individually processed by the trained GAT-based classifier $\mathcal{F}_\Omega$. For each graph G, the classifier outputs a hallucination probability:

$$\hat{y} = \mathcal{F}_\Omega(G) \in [0, 1]. \tag{6}$$

This probability is then thresholded to produce a binary hallucination label $o \in \{0, 1\}$, where $o = 1$ indicates hallucinated content and $o = 0$ indicates factual content. Finally, the probabilities and labels across all paragraphs are aggregated to provide a comprehensive assessment of the entire dialogue instance.

4 EXPERIMENTS

We perform a comprehensive empirical evaluation of SIGHT to validate its effectiveness in identifying hallucinations across diverse domain-specific contexts. Our analysis specifically examines the trade-off between detection accuracy and computational efficiency, benchmarking SIGHT against a broad spectrum of uncertainty estimation, classifier-based, and LLM-based detectors.

4.1 Experiment Settings

Datasets We conducted experiments on two benchmark datasets: MSMARCO-QnA and RAGTruth-QA. For the MSMARCO-QnA dataset, we utilized a processed version released by Nonkes et al. (2024), available from an open-source repository[3]. This dataset comprises 2,000 samples from the biomedical domain, selected from the MSMARCO QnA task[4], where each sample includes a query, an answer, and its supporting context[5]. In addition, derivative responses were generated using large language model under two settings: with and without access to the context[6] [7]. These responses include both factual and hallucinated outputs, with a correct-to-hallucinated ratio of approximately 1:4. We used this dataset to investigate whether incorporating graph-based representations improves hallucination detection performance and to benchmark our method against other graph-based approaches. The data were split into training, validation, and test sets at 70%/15%/15%. Table 1 summarizes the dataset statistics. Furthermore, we used the RAGTruth-QA dataset[8], introduced by Niu et al. (2023). RAGTruth is a large-scale benchmark for assessing hallucination detection in retrieval-augmented generation (RAG) across multiple tasks. We selected a QA task subset to evaluate the generalizability and competitiveness of our method against state-of-the-art hallucination-detection approaches. The dataset was split into training and test sets at 85%/15%. Table 2 summarizes the dataset statistics.

Table 1: **MSMARCO-QnA Dataset Statistics.** Summary of the partitioned biomedical domain corpus.

Category	Component	Count	Total
Source Data	Context Paragraphs	2,000	2,000
	Unique Queries	2,000	2,000
LLM Responses	Factual Samples ($\mathcal{G}^+$)	4,400	
	Hallucinated Samples ($\mathcal{G}^-$)	17,600	22,000
Data Partition	Training Set	15,400	
	Validation Set	3,300	
	Testing Set	3,300	22,000

Table 2: **RAGTruth-QA Dataset Statistics.** Composition of the large-scale benchmark for RAGs.

Category	Component	Count	Total
Source Data	Context Paragraphs	989	989
	Unique Queries	989	989
LLM Responses	Factual Samples ($\mathcal{G}^+$)	4,210	
	Hallucinated Samples ($\mathcal{G}^-$)	1,724	5,934
Data Partition	Training Set	5,044	
	Testing Set	890	5,934

Baselines. To evaluate the *effectiveness* of our proposed method, we conducted comparative experiments using the processed MSMARCO-QnA and RAGTruth-QA datasets, benchmarking against 2 and 6 baselines, respectively. For the MSMARCO-QnA dataset, we compared SIGHT with two baselines: (1) **RoBERTa**, a strong transformer-based text classifier proposed by (Liu et al., 2019), and (2) the **Sentence-based GAT** hallucination detection model proposed by Nonkes et al. (2024), which is structurally similar to our approach. The key differences lie in the granularity of the graph construction, the level of sentences versus our word-level graphs, and the hallucination prediction strategy adopted by the respective models. For the RAGTruth-QA dataset, we compared SIGHT with six baselines representative of three primary categories of hallucination detection approaches: (1) **Uncertainty estimation-based** approach: We compared SelfCheckGPT[9] (Manakul et al., 2023); (2) **Classifier-based** approach: We compared HHEM-2.1-open[10]; and (3) **LLM-based** approach: We compared five LLMs, including GPT-4o[11], GPT-4o-mini[12], GPT-3.5-turbo[13], LLaMA-3.1-8B[14] and Mistral-8B[15].

[3] Github: Hallucination Detection in LLMs
[4] Github: MSMARCO-QnA dataset
[5] Github: Hallucination Detection in LLMs sampled data
[6] Github: Hallucination Detection in LLMs generated with context data
[7] Github: Hallucination Detection in LLMs generated no context data
[8] Github: RAGTruth dataset
[9] Github: SelfCheckGPT
[10] Vectara huggingface: Hallucination evaluation model
[11] OpenAI: GPT-4o
[12] OpenAI: GPT-4o-mini
[13] OpenAI: GPT-3.5-turbo
[14] HuggingFace: LLaMA-3.1-8B
[15] HuggingFace: Mistral-8B

Evaluation Metrics. *Precision*, *recall*, and *F1 score* are used to evaluate the performance of the hallucination detection classification. Additionally, we measure the *inference time (in sec.)* and the *number of model parameters* for each baseline method, to assess the technical considerations relevant for practical deployment. We evaluated all baselines and our method by reporting the mean results over three independent testing runs.

Implementation Details. For training the sentence-based GAT model, we used the open source code[16] provided by Nonkes et al. (2024) and adopted the default training parameters specified in the code. For our proposed method, SIGHT, to investigate the impact of training on the original knowledge text, we used the context corresponding to each query as the training data. In the data preprocessing step, since each context was already a paragraph, we treated the entire context as a single paragraph. We used the context corresponding to each query as the training data. After training, our model was evaluated using the 15% test set. All experiments were performed with a Nvidia A100 GPU. Additional hyperparameters and implementation details are provided in Appendix B.

4.2 Experiment Results

Evaluation Results on MSMARCO-QnA. As shown in Table 3, our method SIGHT outperforms both the RoBERTa method and the sentence-based GAT method. This demonstrates that incorporating graph structures yields better performance compared to using a standard sequential text representation. Furthermore, our newly proposed graph construction and hallucination detection strategy achieves better performance than the sentence-based GAT approach, which is structurally similar to ours. Additionally, by training the detection model solely on the original knowledge context rather than on the responses, we found that the model still retains detection capabilities.

Evaluation Results on RAGTruth-QA. The performance and development-related metrics of our method compared to other benchmark hallucination detection approaches are summarized in Table 4. In terms of detection accuracy, our method outperforms SelfCheckGPT based on uncertainty estimation and the LLM-based methods GPT-3.5-turbo and Mistral-8B. Among all methods, the classifier HHEM-2.1-open attains the best detection performance, followed by the LLM-based GPT-4o-mini and LLaMA-3.1-8B. Regarding model size, our method yields the smallest parameter size, followed by HHEM-2.1-open, while GPT-4o has the largest parameter size. In terms of inference time, HHEM-2.1-open achieves the shortest inference time, followed by our method, with SelfCheckGPT requiring the longest inference time. When considering the need for manual annotation, only HHEM-2.1-open requires human-labeled data for fine-tuning. Based on these evaluation metrics, HHEM-2.1-open is the most suitable choice when prioritizing detection accuracy, fast inference, and moderate model size. However, its reliance on manually annotated data presents a significant limitation. For LLM-based detection methods, the results indicate that stronger LLMs generally yield better detection performance. However, inference speed varies depending on the specific LLM, and additional considerations such as the cost of accessing non-open-source models like OpenAI's offerings must also be taken into account. As for SelfCheckGPT, since it requires generating multiple samples for comparison and verification, it incurs the longest inference time, making it less suitable for real-world applications that demand rapid response times.

[16] Github: Hallucination Detection in LLMs

Table 3: **Detection performance on MSMARCO-QnA.** The ↑ indicates relative improvement in F1 (%) over the **S-GAT** (Sentence-based GAT) baseline (Nonkes et al., 2024). Best results are in **bold**.

Method	Precision (%)	Recall (%)	F1 (%)	ΔF1 (%)
RoBERTa	48.74 ± 0.32	66.67 ± 0.00	56.31 ± 0.21	+3.36%
S-GAT	58.84 ± 2.60	50.80 ± 2.00	54.48 ± 0.01	–
SIGHT	**74.19** ± 0.25	**86.91** ± 1.47	**80.05** ± 0.86	**+46.94%**

Efficiency-Performance-Size Comparison. Figure 4 compares SIGHT with state-of-the-art detectors along accuracy, parameter count, and inference time. Although SIGHT does not achieve the highest accuracy, it lies in a favorable region of the Pareto frontier by combining label-free training with a compact detector and single-pass scoring. In practice, this yields competitive detection performance with substantially lower latency and a smaller footprint than retrieval-based or auxiliary LLM approaches. The elimination of annotation cost, combined with the small model size and fast inference, makes SIGHT well-suited for resource-constrained deployments.

5 ABLATION STUDY

Hallucinated Data Generation. To analyze the contribution of the self-supervised learning strategy in the SIGHT framework, we conducted an ablation study by removing the LLM-based hallucinated data generation module. As illustrated in Figure 5, the absence of LLM-generated hallucinations led to a notable decline in detection performance. This confirms that incorporating LLM-generated hallucinated texts significantly enhances the model's ability to identify

Table 4: **Comparative performance on RAGTruth-QA.** Detection results across various architectural paradigms, including uncertainty estimation, LLM-based prompting, and specialized classifiers. The ↓ indicates degradation in F1 (%) compared to the supervised HHEM-2.1-open baseline. Best results are highlighted in **bold**.

Method	Approach	Annot.	Params	Inference (s)	Precision (%)	Recall (%)	F1 (%)	ΔF1 (%)
SelfCheckGPT-NLI	Uncertainty	✗	435.06M	64.462 ± 0.267	33.18 ± 9.40	62.10 ± 5.15	42.96 ± 9.31	↓ 30.55%
GPT-4o	LLM based	✗	200B	1.208 ± 0.099	38.71 ± 5.33	85.34 ± 1.20	53.09 ± 4.78	↓ 14.16%
GPT-4o-mini	LLM based	✗	8B	0.996 ± 0.065	51.25 ± 4.53	51.37 ± 0.20	51.24 ± 2.17	↓ 17.15%
GPT-3.5-turbo	LLM based	✗	20B	0.987 ± 0.095	25.12 ± 6.07	**89.85 ± 8.36**	39.15 ± 8.24	↓ 36.69%
LLaMA-3.1-8B	LLM based	✗	8B	2.412 ± 0.646	32.36 ± 7.40	87.68 ± 1.98	46.99 ± 8.36	↓ 24.05%
Mistral-8B	LLM based	✗	8B	0.615 ± 0.181	29.72 ± 4.90	42.14 ± 2.35	34.58 ± 2.82	↓ 44.08%
HHEM-2.1-open	Classifier	✓	110M	**0.014 ± 0.002**	**75.69 ± 8.12**	52.43 ± 1.52	**61.85 ± 3.38**	—
SiGHT (ours)	Classifier	✗	**0.03M**	0.341 ± 0.003	35.25 ± 8.20	68.14 ± 3.92	45.90 ± 6.90	↓ 25.79%

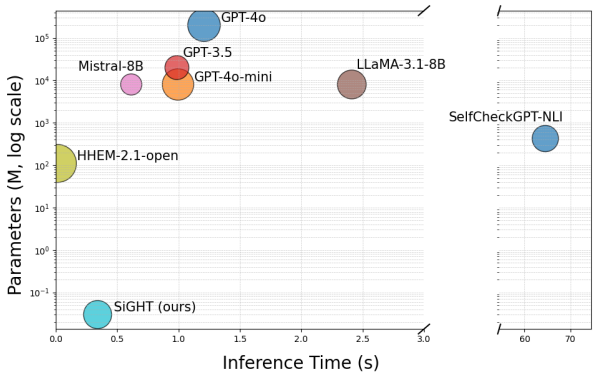

Figure 4: Efficiency, performance, and size comparison of SiGHT against state-of-the-art detectors. Each point denotes a method; the x-axis reports inference time per sample, the y-axis reports parameter count, and dot size encodes F1 score (larger is better). SiGHT achieves a favorable trade-off between low latency and a small footprint, while maintaining a competitive F1.

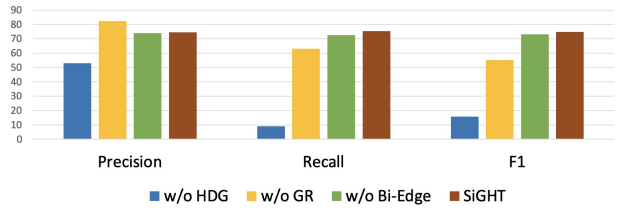

Figure 5: **Ablation study results.** Performance impact of key framework components: (w/o HDG) denotes the removal of LLM-based hallucinated data generation; (w/o GR) represents the substitution of the graph-based representation with a standard text classification baseline; and (w/o Bi-Edge) indicates the exclusion of bidirectional edges from the word-level graph topology.

hallucinations. The improvement stems from the fact that LLM-generated hallucinations are linguistically fluent and closely resemble human-written text, thereby creating more challenging and realistic training samples that better reflect the complexity of real-world hallucination scenarios.

Fine-grained Relational Semantics. To evaluate the efficacy of modeling fine-grained relational semantics, we compare our graph-based architecture against a standard text classification baseline. As illustrated in Figure 5, replacing the graph structure with a linear text encoder leads to a marked decrease in both Recall and F1 score. This performance drop validates the importance of SiGHT's structure-aware learning, which captures lexical inconsistencies that remain latent in sequence-only models. Furthermore, our high-resolution approach demonstrates a 46.94% F1 improvement over the coarse-grained, sentence-level framework proposed by Nonkes et al. (2024) (Table 3), highlighting the necessity of token-level granularity for nuanced hallucination detection.

Bidirectional Edge Design. We further examine the influence of topological design by ablating bidirectional edges within the word-level graph. Specifically, we compare the full SiGHT model against a variant restricted to unidirectional edges following the original lexical sequence. The results, shown in Figure 5, indicate that the removal of bidirectional connectivity consistently degrades Precision, Recall, and F1 scores. This suggests that bidirectional edges allow the graph attention mechanism to better model mutual contextual dependencies and non-sequential associations between neighboring tokens, which are frequently disrupted in hallucinated outputs.

6 RELATED WORK

The escalating difficulty of identifying LLM-generated misinformation (Chen and Shu, 2023) has catalyzed the development of diverse hallucination-detection methodologies (Huang et al., 2025). For domain-specific con-

texts, these approaches are generally categorized into three families: uncertainty estimation, classifier-based detectors, and LLM-based verification.

Uncertainty Estimation. These methods assume that hallucinations correlate with uncertainty in the states or behaviors of the model (Huang et al., 2025). For example, SAPLMA (Azaria and Mitchell, 2023) trains a classifier on hidden activations to predict factuality, presuming that internal states encode logical and factual signals; the model relies on human-annotated true and false statements in multiple topics. INSIDE (Chen et al., 2024b) introduces the EigenScore from the activation covariance and applies feature clipping at test time to temper overconfident errors. When internal states are unavailable, the behavioral uncertainty is estimated from the variation in the output. Semantic entropy (Farquhar et al., 2024) groups multiple responses for a prompt and computes the entropy over the assignment of the clusters, while SelfCheckGPT (Manakul et al., 2023) also samples multiple outputs, but evaluates consistency using an LLM or an NLI model rather than entropy. The main limitations are the requirement for internal access or confidence signals and the computational cost associated with repeated sampling. By contrast, our approach trains a small external detector on structured knowledge, removing both internal access and multi-sample inference.

Classifier-based Detectors. These approaches learn a direct discriminator between factual and hallucinated content from curated data. Textual detectors such as the widely adopted Hughes Hallucination Evaluation Model (HHEM)[17][18] cast detection as natural language inference using a cross-encoder trained on annotated premise-hypothesis pairs, and achieve strong results across benchmarks. However, they depend on large human-labeled corpora, which limits scalability. Structure-aware detectors encode relationships explicitly. Knowledge graph-based methods (Kim et al., 2023) enrich input with curated graph structure, for example, GAT over KG augmented text (Furumai et al., 2023) or KG plus NLI for verification (Sansford et al., 2024). These methods rely on costly graph construction and curation. To reduce manual effort, Nonkes et al. (2024) replaces external KGs with sentence-level graphs in latent space and trains a GAT to classify responses, but the coarse granularity of sentences can miss subtle token-level inconsistencies. In short, classifier-based work faces two challenges: annotation dependence and coarse structure. Our framework addresses both by generating training data automatically with prompt guidance and by operating on word-level graphs that expose fine-grained relational cues.

LLM-based Verification. LLM-based verification constitutes an alternative paradigm in which models serve as direct evaluators of faithfulness (Chiang and Lee, 2023; Huang et al., 2025). For instance, retrieval-enhanced verification (Huo et al., 2023) validates responses against external evidence, while the Chain of Verification (CoV) (Dhuliawala et al., 2023) framework employs a multi-stage protocol of generation, retrieval, and explanation to refine output accuracy. Similarly, SelfCheckGPT (Manakul et al., 2023) reconciles model uncertainty with LLM-based judgment by assessing consistency across sampled outputs via prompting or Natural Language Inference (NLI). Although effective when supported by high-capacity models, these methodologies incur substantial computational overhead, exhibit sensitivity to prompt engineering, and depend on external retrieval systems. In contrast, SIGHT bypasses inference-time LLM queries and retrieval, enabling high-throughput classification via a compact, efficient graph encoder.

7 CONCLUSION

We present SIGHT, a self-supervised graph-based architecture for hallucination detection in domain-specific Large Language Models. By synthesizing counterfactual training data from structured knowledge, the framework eliminates the requirement for manual annotation while capturing fine-grained relational dependencies through high-resolution word-level graphs. At inference, SIGHT enables efficient single-pass classification without requiring external retrieval, auxiliary LLM calls, or internal model access. Evaluations on MSMARCO-QnA and RAGTruth-QA benchmarks demonstrate that SIGHT substantially outperforms existing graph-based methods and achieves competitive results against state-of-the-art detectors. With only 0.03M parameters and sub-second latency, SIGHT provides a scalable and practical solution for maintaining factual integrity in high-stakes, resource-constrained environments. Future research will scale SIGHT for long form evaluation, optimizing graph memory efficiency while preserving structural coherence across extended contexts.

Acknowledgements

This work is partially supported by the National Science and Technology Council (NSTC), Taiwan (Grants: 114-2222-E-A49-004, and 114-2639-E-A49-001-ASP). Additional support was provided by National Yang Ming Chiao Tung University (NYCU) and the Institute for Information Industry (III).

[17] Vectara HHEM v2: a new and improved factual consistency scoring model

[18] Vectara huggingface: hallucination evaluation model

References

Desta Haileselassie Hagos, Rick Battle, and Danda B Rawat. Recent advances in generative ai and large language models: Current status, challenges, and perspectives. *IEEE Transactions on Artificial Intelligence*, 2024.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 2025.

SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*, 2024.

Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. Llm-check: Investigating detection of hallucinations in large language models. *Advances in Neural Information Processing Systems*, 2024.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, et al. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv preprint arXiv:2310.07521*, 2023.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024a.

Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*, 2024.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning llms on new knowledge encourage hallucinations? *arXiv preprint arXiv:2405.05904*, 2024.

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. *arXiv preprint arXiv:2304.13734*, 2023.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*, 2023.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*, 2023.

Siqing Huo, Negar Arabzadeh, and Charles LA Clarke. Retrieving supporting evidence for llms generated answers. *arXiv preprint arXiv:2306.13781*, 2023.

Kazuaki Furumai, Yanan Wang, Makoto Shinohara, Kazushi Ikeda, Yi Yu, and Tsuneo Kato. Detecting dialogue hallucination using graph neural networks. In *2023 International Conference on Machine Learning and Applications (ICMLA)*, 2023.

Hannah Sansford, Nicholas Richardson, Hermina Petric Maretic, and Juba Nait Saada. Grapheval: A knowledge-graph based llm hallucination evaluation framework. *arXiv preprint arXiv:2407.10793*, 2024.

Noa Nonkes, Sergei Agaronian, Evangelos Kanoulas, and Roxana Petcu. Leveraging graph structures to detect hallucinations in large language models. *arXiv preprint arXiv:2407.04485*, 2024.

Kenneth Ward Church. Word2vec. *Natural Language Engineering*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, Kashun Shum, Randy Zhong, Juntong Song, and Tong Zhang. Ragtruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. *arXiv preprint arXiv:2401.00396*, 2023.

Marc Pouly et al. Estimating text similarity based on semantic concept embeddings. *arXiv preprint arXiv:2401.04422*, 2024.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. 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)*, 2019.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019.

Zhilin Wang, Yafu Li, Jianhao Yan, Yu Cheng, and Yue Zhang. Unveiling attractor cycles in large language models: A dynamical systems view of successive paraphrasing. *arXiv preprint arXiv:2502.15208*, 2025.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2017.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Canyu Chen and Kai Shu. Can llm-generated misinformation be detected? *arXiv preprint arXiv:2309.13788*, 2023.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. Inside: Llms' internal states retain the power of hallucination detection. *arXiv preprint arXiv:2402.03744*, 2024b.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 2024.

Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. Factkg: Fact verification via reasoning on knowledge graphs. *arXiv preprint arXiv:2305.06590*, 2023.

Cheng-Han Chiang and Hung-yi Lee. Can large language models be an alternative to human evaluations? *arXiv preprint arXiv:2305.01937*, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

SiGHT: A Self-Supervised Graph-based Hallucination DeTection Framework for Domain-Specific LLMs: Supplementary Materials

A HALLUCINATION DATA GENERATION PROMPT

A.1 Prompt Template

Figure 6 depicts the prompt template for generating hallucinated text.

You are a professional hallucinated text generator. Your task is to generate incorrect but natural texts based on the provided correct texts and related words.

Given an original text and a set of specific words along with their top 5 related words, replace each specific word in the original text with one of its related words.

If there are no related words suitable for replace specific word, keep the original word.

The final generated hallucinated text must:
1. Maintain a grammatically correct and natural structure.
2. Use the provided related words for replacement.
3. Change the meaning of the original text, making it incorrect.
4. Still appear plausible as a valid text.

The output format must be a single hallucinated string. If no hallucinated string can be generated, return 'None' string.
Just return the hallucinated text generated from original text, don't return previous text!
Remind again! Please don't output any other information (like code) except hallucinated text or "None" string, or I will punish you!

Input:
Original text:
"Numbness in a body part refers to a loss of sensation or feeling in that area, often caused by nerve damage or compression."

Related words:
- "nerve" : "tissue", "ligament", "filament", "sinew", "cord"

Output:

Figure 6: The complete prompt template ($\texttt{PT}(\cdot)$) used for generating hallucinated texts. The prompt guides the LLM ($\text{LLM}_{gen}(\cdot)$) to produce factually incorrect but fluent texts based on the factual domain knowledge. The prompt consists of three inputs: The factual text serves as the knowledge source (p). The top-N similar terms ($\mathcal{N}(w)$) are provided to constrain the substitution vocabulary. The value of N indicates the count of similar terms provided.

A.2 Instructions and Design Rationales for Hallucinated Data Generation

To ensure the generated hallucinated data is challenging to detect, we designed a prompt template presented in Figure 6 that incorporates four core instructions. These instructions guide the Large Language Model to synthesize content that is semantically coherent but factually incorrect. The objectives of these instructions

are to simulate realistic hallucination error types from various dimensions, compelling our detection model to effectively capture subtle structural inconsistencies rather than merely surface-level errors. Table 5 details the four instructions and their specific design rationales within the SiGHT framework.

Table 5: Detailed Instructions and Design Rationales for Hallucinated Data Generation.

Instruction Content	**Design Rationale**
Change the meaning of the original text, making it incorrect	**Ensuring Factual Error:** To guarantee the generated content conflicts with the true facts or domain knowledge, providing the core negative signal for the detection framework.
Still appear plausible as a valid text.	**Maintaining Plausibility and Deceptiveness:** To simulate subtle hallucinations. This forces the generated text to maintain high fluency and deceptive quality, increasing the difficulty for the detector.
Maintain a grammatically correct and natural structure.	**Guaranteeing Grammatical Correctness:** To prevent the model from distinguishing based on surface-level syntax errors, compelling it to learn fine-grained relational semantics.
Use the provided related words for replacement.	**Controlling Error Scope and Type:** To focus the error type. This restricts the LLM to generate targeted hallucinations by substituting key terms with semantically similar, but factually false, entities.

B HYPERPARAMETER SETTINGS

SiGHT Model Training Details. We segmented sentences using punctuation symbols ('.', '?', '!', and \n) and tokenized words by spaces. We filtered out 12 parts of speech considered to be non-meaningful: subordinating conjunctions, particles, punctuation marks, whitespace characters, symbols, interjections, coordinating conjunctions, determiners, adverbs, pronouns, adpositions, and miscellaneous or unknown tokens. For MSMARCO-QnA, we trained a Word2Vec model (vector size d=100, skip-gram, 400 epochs, sampling rate 1e-4, min count=one, window=ten). In the word graph generation step, we used a prompt template $PT()$, as shown in the Appendix A.1. In the template, the top ten related words were provided, and one negative text was generated per prompt. We set the temperature parameter of the LLM to 1.0, while keeping all other parameters at their default values. In the detection model training step, we set the training parameters as follows: 60 epochs, batch size 16, learning rate 1e-6, AdamW optimizer, four attention heads, and a hidden dimension size of 32. For RAGTruth-QA, we trained a Word2Vec model (vector size d=100, skip-gram, 200 epochs, sampling rate 1e-4, min count=one, window=ten). In the word graph generation step, we used the prompt template $\texttt{PT}(\cdot)$ shown in Appendix A.1, where the top ten related words were provided and a negative text was generated per prompt. We set the temperature parameter of the LLM to 1.0, while keeping all other parameters at their default values. In the detection model training step, we set 60 epochs, a batch size of 16, a learning rate of 1e-5, an AdamW optimizer, six attention heads, and a hidden dimension size of 16.

Additional Implementation Details. In the experiments conducted on the MSMARCO-QnA dataset, since no official train–test split is provided, we constructed three independent train/validation/test splits by random resampling. All results reported in Table 3 are averaged over these three splits. For each split, the training and validation sets were used for model training and hyperparameter tuning, respectively. Both the RoBERTa classification model and the sentence-based GAT model were trained under identical conditions for fair comparison. For the RoBERTa model, we applied a standard preprocessing pipeline that included whitespace normalization, lowercase, and stop-word removal based on the NLTK dictionary. The preprocessed text was then tokenized using the RoBERTa tokenizer before being fed into the model. The model was trained with a batch size of 16, 20 epochs, a learning rate of 0.001, and an SGD optimizer. In experiments using the RAGTruth-QA dataset, we directly adopted the split training and test set provided in the RAGTruth QA task dataset. All baselines were evaluated using the same 15% test set. For the SelfCheckGPT experiment, we generated four samples per response and performed hallucination detection using the NLI method recommended by SelfCheckGPT, implemented

Table 6: Performance on MSMARCO-QnA dataset using different sizes of LLMs to generate hallucinated text. The ↑ indicates improvement in F1 (%) of the 8B model over the 70B model. The ↓ indicates degradation in Accuracy (%) of the 8B model relative to the 70B model.

LLMs	MSMARCO-QnA				MSMARCO-QnA Sample Data	
	Precision	Recall	F1	ΔF1 (%)	Accuracy	ΔAccuracy (%)
LLaMA 3.1-8B	74.44	**75.40**	**74.91**	↑ 9.43%	32.43	↓ 41.10%
LLaMA 3.3-70B	**75.25**	62.80	68.46	—	**55.07**	—

through the official selfcheckgpt package[19]. For prompt verification experiments, GPT-4o, GPT-4o-mini, and GPT-3.5-turbo models were accessed through the OpenAI API[20] [21] [22]. In contrast, LLaMA 3.1-8B and Mistral-8B models were downloaded from HuggingFace[23] [24] and used for local prompting. We used the QA prompt template provided in RAGTruth (Niu et al., 2023) for prompt verification. The number of model parameters for OpenAI models is derived from two public sources[25] [26]. For HHEM-2.1-open experiments, we utilized the open-source code[27] available on HuggingFace for hallucination detection.

C PARAMETRIC-EFFICIENCY STUDY

Setup. To investigate the impact of LLM size on hallucination detection performance, we conducted experiments using two variants of the LLaMA3 (Grattafiori et al., 2024) model: LLaMA3-8B and LLaMA3-70B, with training data generated from each model. All evaluations were performed on the MSMARCO-QnA dataset.

Results. As shown in Table 6, detection models trained on hallucinated texts generated by the smaller LLM (8B) exhibited superior performance in identifying hallucinated content, reflected by higher F1 scores. However, when tested exclusively on factual inputs (i.e., all hallucination labels set to 0), the model trained with data from the larger LLM (70B) showed stronger accuracy in correctly identifying non-hallucinated responses. Interestingly, we also observed that prompting the smaller LLM required more stringent formatting and explicit instruction design; otherwise, it frequently failed to follow the intended generation behavior. In contrast, the larger LLM was more robust to prompt variation and generated hallucinations with greater consistency. These results reveal a nuanced trade-off between the hardness of negative samples and the reliability of the label. Smaller LLMs tend to generate more diverse and challenging counterfactuals, which can strengthen the robustness of the detector. Still, they also increase the risk of corrupting factual content and require tighter control over prompts. In contrast, larger LLMs achieve higher precision on non-hallucinated text, likely due to stronger grounding and instruction following; yet, their negatives are often milder and less informative. Going forward, we advocate adaptive prompting that tunes temperature and perturbation scope by model size, and a hybrid generation pipeline that mixes negatives from multiple LLMs with automatic quality filters and agreement checks, thereby balancing hardness and fidelity in the training data.

D PARAMETRIC-SENSITIVITY STUDY

Setup. To further examine the impact of hallucinated data generation parameters, we conducted a sensitivity analysis on the temperature setting of the LLM used for negative data synthesis. Temperature directly controls the diversity and randomness of the generated hallucinated text, thereby affecting both the hardness and reliability of training data. All experiments are conducted on the MSMARCO-QnA dataset, while keeping other generation settings fixed.

[19] Github: SelfCheckGPT
[20] OpenAI: GPT-4o
[21] OpenAI: GPT-4o-mini
[22] OpenAI: GPT-3.5-turbo
[23] HuggingFace: LLaMA-3.1-8B
[24] HuggingFace: Mistral-8B
[25] https://www.communeify.com/tw/blog/chatgpt-series-compare
[26] https://aclanthology.org/2023.emnlp-main.716/
[27] Vectara huggingface: Hallucination evaluation model

Table 7: Performance on MSMARCO-QnA dataset using different temperature setting of the LLM to generate hallucinated text.

Temperature	Precision	Recall	F1
0.3	72.77	68.16	69.28
0.7	**74.40**	75.23	74.81
1.0	74.19	**86.91**	**80.05**

Results. As shown in Table 7, increasing the temperature consistently improves Recall and F1 score, with the best performance achieved at temperature 1.0. In particular, raising the temperature from 0.3 to 1.0 results in a substantial F1 improvement, indicating that more diverse hallucinated data significantly enhance ability of the detector to capture subtle semantic inconsistencies. However, we observe that lower temperature settings produce more conservative hallucinations with limited lexical and semantic variation, which reduces the effectiveness of contrastive learning. In contrast, higher temperature values encourage broader perturbations and introduce more challenging counterfactuals, leading to stronger generalization. These findings suggest that temperature plays a critical role in balancing hallucination diversity, and should be carefully tuned when generating negative data.

E LIMITATIONS

Our approach has three main limitations. **Quality of negatives.** The fidelity of $\mathcal{G}^-$ depends on the underlying LLM and the prompt template;. However, we quantify coverage and soundness on a labeled subset; residual bias may persist under domain shift. **Sequence length and memory.** Token-level graphs raise memory usage as sequences grow; sparse connectivity and block attention mitigate the cost, but extreme lengths remain challenging. **Scope.** The present study targets textual hallucinations; extending the detector to multimodal inputs such as tables and figures is left for future work.