Neurosymbolic Artificial Intelligence
XX(X):1–28

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

KEA Explain: A Neurosymbolic Framework for Explaining LLM Hallucinations

Reilly Haskins and Benjamin Adams

Abstract

Large Language Models (LLMs) frequently generate hallucinations: statements that are syntactically plausible but lack factual grounding. This research presents KEA (Kernel-Enriched AI) Explain, a neurosymbolic framework that detects and explains such hallucinations by comparing knowledge graphs constructed from LLM outputs with ground truth data from Wikidata or contextual documents. Using graph kernels and semantic clustering, the method provides explanations for detected hallucinations, ensuring both robustness and interpretability. Our framework achieves competitive accuracy in detecting hallucinations across both open- and closed-domain tasks, and is able to generate contrastive explanations, enhancing transparency. This research advances the reliability of LLMs in high-stakes domains and provides a foundation for future work on precision improvements and multi-source knowledge integration.

Keywords

large language models, hallucination, knowledge graph, neurosymbolic, graph kernel, explainability

Introduction

Despite the impressive capabilities of Large Language Models (LLMs), they frequently generate outputs that are grammatically correct but semantically incorrect—referred to

Department of Computer Science and Software Engineering, University of Canterbury, New Zealand

Corresponding author:
Reilly Haskins
Email: reilly.haskins@pg.canterbury.ac.nz

as hallucinations (Huang et al. 2025). These hallucinations are a significant challenge, particularly when language models are applied in high-stakes domains such as healthcare, legal advice, and education, where misleading information can seriously degrade user trust (Oelschlager 2024). Thus, developing methods to detect and explain hallucinations is critical to ensuring the reliability and trustworthiness of LLMs.

Methods for detecting hallucinations fall under one of two categories: closed-domain and open-domain (Chen and Yih 2020). Closed-domain hallucination detection refers to detection of hallucinations using a specific constrained context which is supplied alongside the LLM's generated content. This context can take the form of a document, dataset, or passage of text. In the closed-domain, the focus is on whether the generated content aligns well with a specific piece of knowledge. In contrast, open-domain hallucination detection is a class of problem that aims to detect hallucinations without being constrained to a specific, predefined body of knowledge. In the open-domain, the focus is on determining whether the generated content aligns with general knowledge or truth.

In this article, we propose a neurosymbolic framework, KEA (Kernel-Enriched AI) Explain, that achieves performance on par with state-of-the-art methods for detecting semantic hallucinations in LLM-generated text under both open-domain (general knowledge) and closed-domain (constrained context) settings, while offering the key advantage of providing explanations for why statements are deemed hallucinatory. By integrating symbolic AI methods with state-of-the-art neural techniques, our approach combines the strengths of both paradigms to address key limitations in existing methods. Specifically, we use graph kernels (Vishwanathan et al. 2010) to compare the structural similarity between knowledge graphs (KGs), while semantic word embeddings (Almeida and Xexéo 2019) are employed to cluster semantically similar labels and select relevant triples. For the open-domain problem, these techniques are combined with structured knowledge from Wikidata (Vrandečić and Krötzsch 2014), a comprehensive and widely-used knowledge base, to form a robust detection system. We extract entities and relations from the LLM output and query these against the knowledge base, systematically identifying and explaining hallucinations based on discrepancies in structural and semantic similarity. For the closed-domain problem, we construct a KG based on both the LLM output and the provided context in place of the Wikidata knowledge. There is evidence that people prefer explanations that are contrastive; that is, they explain why one outcome occurred rather than another plausible alternative (Miller 2019). By providing *why* a given output was classified as a hallucination while also providing what fact would make it *not* a hallucination, KEA Explain offers a more explainable and intuitive solution.

The key contributions of this work are as follows:

- Introduces the use of graph kernels over symbolic knowledge graphs to detect and explain hallucinations in LLM outputs.
- Presents a novel neurosymbolic framework to enable comparison of semantic alignment between entity and relation labels of a knowledge graph pair.
- Improves upon existing methods by introducing explainable classifications, which are driven by the symbolic structure of the graph-based representation.

Background and Related Work

In this section, we outline the key concepts that underpin our approach. We begin with knowledge graphs, which provide the structural foundation for representing and comparing factual information. We then introduce agglomerative hierarchical clustering, a core technique used for semantic alignment in our framework. Finally, we discuss recent work on detection of hallucinations in large language models (LLMs), which frame the central challenge our method is designed to address.

Knowledge Graphs

A knowledge graph (KG) is a structured representation of information, where entities are stored as nodes, and relationships between them are represented as directed edges (Hogan et al. 2021). The information in a KG is often represented as triples in the form (h,r,t), where h and t are entities, and r represents the relationship linking them e.g., ("Albert Einstein", "was born in", "Ulm") (Chen et al. 2020). This structure allows KGs to provide a relational and interpretable view of data, enabling inference of knowledge effectively, which we rely on heavily in this project to compare information-rich textual data. The ability of KG structure makes detected inconsistencies between multiple graphs localizable, which we later develop into human-understandable explanations.

Agglomerative Hierarchical Clustering

Hierarchical clustering builds a hierarchy of clusters by merging or splitting data points according to a similarity metric. In the agglomerative approach (AHC), each data point begins in its own cluster, and clusters are successively merged based on a distance metric (Murtagh and Contreras 2012). The process continues until all points belong to a single cluster or until a stopping criterion is reached, typically a minimum distance threshold.

The similarity between clusters is determined using the distance metric, such as Euclidean distance or cosine similarity, and a linkage criterion, which defines how the distances between clusters are calculated. In our work, we use average linkage as the criterion, where we consider the average pairwise distance between *all* points in the clusters.

AHC has several advantages over alternative clustering methods, such as k-means. It does not require a pre-defined number of clusters, does not assume clusters of a specific size or shape, and is particularly useful for data containing nested structures, such as graphs and taxonomies. However, it is computationally intensive for large amounts of data, as the algorithm scales quadratically with the number of data points.

Graph Kernels

Graph kernels are a family of similarity functions that allow machine learning methods to operate on graphs by comparing their structural patterns numerically (Vishwanathan et al. 2010). They work by mapping graphs into a high-dimensional feature space, where similarities can be quantified in terms of shared substructures such as walks, paths,

or subtrees. This makes them particularly useful for tasks where relational structure is important, such as in knowledge graphs.

A wide range of graph kernels have been proposed, each emphasizing different aspects of graph structure. In this work, we make use of the Weisfeiler-Lehman (WL) subtree kernel (Shervashidze et al. 2011), which is widely adopted due to its efficiency and strong empirical performance. The WL kernel captures similarity between graphs by iteratively refining node labels based on their neighborhoods, effectively encoding increasingly rich structural patterns at each iteration. Its balance between computational efficiency and expressive power makes it particularly well suited for comparing knowledge graphs derived from natural language. Compared with triple-wise checks, WL captures subtree context, which later lets us point to minimal graph edits rather than opaque scores.

LLM Hallucinations

Recent advances in natural language generation have been largely driven by deep learning models, in particular transformer-based architectures used for large language models (LLMs). These models have revolutionized tasks such as text generation and machine translation, but alongside these successes, researchers and practitioners have become increasingly aware of their limitations around reliably producing factual outputs (Ji et al. 2023; Tonmoy et al. 2024). Hallucinations in LLMs have been identified to come from four primary sources: biased training data, lack of in-built logic, vague prompts, and insufficient data / overfitting (Perković et al. 2024). Mitigation techniques include fine tuning and Retrieval-Augmented Generation (RAG) (Gao et al. 2023; Arslan et al. 2024), where a kind of safety net can be implemented within the LLM by providing current, relevant, or previously unexplored data during text generation in the hope that this will counter-act uncertainty in the text generation process and steer the LLM away from producing hallucinations. A downside of RAG as a mitigation technique is that the relevancy of the retrieved documents is heavily dependent on the prompt or question posed to the LLM. As a result, RAG may retrieve sources that are relevant enough to answer the question, but not necessarily sufficient to provide the broader context needed to prevent hallucinations. This implies that while RAG can guide the model toward answering the query, it does not guarantee the retrieval of contextually rich sources that could help avoid errors or hallucinated content (Guan et al. 2024).

The persistent issue of hallucinations is particularly concerning in high-stakes domains such as healthcare, legal systems, and education, where inaccurate or misleading information can have severe consequences in decision-making processes. Addressing hallucinations is therefore not only a technical challenge but also a critical step toward ensuring the reliability and trustworthiness of LLMs in real-world applications.

LLM-based Hallucination Detection Techniques

Many recently developed methods use LLMs or probabilistic measures to detect hallucinations. ChainPoll (Friel and Sanyal 2023) detects hallucinations by polling multiple instances of an LLM to arrive at an aggregated vote on whether the text in question is hallucinatory. Although ChainPoll shows good hallucination detection

accuracy, its reliance on prompt engineering and repeated LLM queries can make it computationally expensive compared to simpler methods, limiting its applicability in resource-constrained environments. Additionally, there is no ground-truth resource, so biases in the training data of the LLMs being used could propagate through when they are unable to detect hallucinations.

Another LLM-based method, SelfCheckGPT (Manakul et al. 2023), uses the entropy of a sample of model responses to a query (with temperature > 0) to arrive at a probability of hallucination, hence it can be considered an open-domain method for hallucination detection. This is based on the hypothesis that if an LLM has knowledge of a certain domain, then its responses are likely to be more homogeneous and contain the same key facts. If instead the LLM has little knowledge of a topic, it is assumed to be more likely to have noisy responses which diverge and contradict each other. SelfCheckGPT achieves impressive performance for a simple method, but is also prone to the same drawbacks as ChainPoll around lack of ground truth, as well as being computationally heavy due to the need for generation of a number of LLM responses to each prompt. Additionally, as a result of this lack of grounding, both of these methods are unable to provide a comprehensive interpretation of exactly *why* a certain example is classified as a hallucination.

Knowledge Graphs for Hallucination Detection and Prevention

Open-Domain. A prominent open-domain knowledge graph-based hallucination detection method is AlignScore (Zha et al. 2023). This method utilizes a unified information alignment function, a model designed to assess the alignment between two pieces of text, and is trained on a diverse dataset spanning seven language tasks, including paraphrasing, fact verification, and summarization.

This extensive training, amounting to 4.7 million examples, enhances AlignScore's ability to evaluate diverse factual inconsistency scenarios. AlignScore outperformed many existing metrics across a wide range of language tasks, but lacks interpretability and explainability. Additionally, AlignScore is trained on synthetic data in order to expose the model to enough context in each domain. While usage of synthetic data increases the model's performance, it introduces questions around generalizability to the real world.

Knowledge Graph-based Retrofitting (KGR) (Guan et al. 2024) is another open-domain framework designed to reduce factual hallucinations in LLMs by refining outputs using knowledge graphs. Unlike previous methods that only retrieve facts based on user queries, KGR is able to extract, validate, and correct facts within the LLM's reasoning process, enabling it to catch inaccuracies generated during intermediate reasoning. The KGR method iteratively performs claim extraction, entity detection, fact selection, claim verification, and response retrofitting, improving LLM accuracy on complex question-answering benchmarks. However, KGR's performance is bottlenecked by the entity detection and fact selection stage, as irrelevant entities or noisy triples can prevent accurate claim validation. This shows the need for further refinement of these components to enhance KGR's effectiveness across diverse reasoning tasks.

Notably, both AlignScore and KGR methodologies are designed to produce an opaque alignment score, and KGR's multi-stage pipeline does not expose the specific relations which drive corrections.

Closed-Domain. FactAlign (Rashad et al. 2024) constructs a KG from a given source and generated text, and uses word embeddings to align individual claim triples with source triples to identify factual misalignments, categorizing them as hallucinations if the similarity is low. FactAlign differentiates between intrinsic hallucinations (distortions of source information) and extrinsic hallucinations (unsupported additions) by employing a contradiction score from a natural language inference (NLI) model. The approach achieves high accuracy scores without requiring training or fine-tuning. However, it excludes non-named entities in the KG, which reduces coverage, and encounters scalability challenges due to the computational overhead of computing pairwise similarities between each generated triplet and all source triplets. Additionally, the triple-level comparison methodology in Rashad et al.'s approach prevents wider context from the knowledge graph from contributing to similarity calculations, a limitation that our graph kernel-based approach overcomes.

GraphEval (Sansford et al. 2024) aims to detect hallucinations under closed-domain conditions by first constructing a KG from the LLM's output and then decomposing information into entities and relationships. Each triple is then compared against the provided grounding context using natural language inference (NLI) models to assess consistency. Experimental results show that the use of GraphEval improves the balanced accuracy of hallucination detection on established benchmarks (SummEval, QAGS-C, and QAGS-X), particularly for longer outputs where traditional methods struggle to maintain consistency across multiple facts. For shorter texts, GraphEval's impact is less pronounced, as fewer facts require verification. A major limitation of this method is that it only focuses on the closed-domain hallucination detection problem, where the method is provided with the output alongside both the original prompt and the contextual information that was used to arrive at that output. This limits generalizability of the method, as in real-world detection problems the prompt and context are not readily available. Additionally, as the method focuses on individual triple pairs, the comparison process cannot take into account the broader structural differences of the KG.

Explainable AI

With the wide adoption of complex machine learning and deep learning systems in various sectors, the field of explainable AI (XAI) is quickly growing. XAI aims to help users understand how decisions from these complex models are made, along with the risks of using them, which holds great benefits for crucial industries. For example, in industries such as healthcare, finance, and education, inaccurate predictions from these models have the potential to lead to serious consequences, sometimes with human life on the line. Hence it is crucial to understand how these systems make their decisions through provision of explanations to their users (Dwivedi et al. 2023).

Work by Miller (2019) explores how insights from philosophy, psychology, and social science can enhance explainability in AI. Miller argues that, despite the recent rise in

popularity of XAI (Arrieta et al. 2020), current methods often overlook contributions from these fields regarding how humans understand explanations, which can lead to poor alignment with user expectations. Key findings include that people naturally seek explanations by comparing actual events with counterfactual alternatives, and that explanations are selective and influenced by cognitive biases, as individuals tend to identify only a few causes from many possibilities. These insights suggest that XAI could benefit from a greater emphasis on contrastive explanations that account for common cognitive biases, in order to create more intuitive explanations, although implementing such contrastive and socially-aware models remains challenging. For this reason, in our work we adopt a counterfactual style of explanation (see Section "Generation of Explanations").

Summary

Across these studies, three common limitations emerge:

1. **Generalizability**: The majority of methods are designed to rely on specific open- or closed-domain conditions, but not both. In addition, many make use of synthetic training data, limiting applicability beyond their training domain.
2. **Lack of ground-truth**: Many of these methods use proxies for ground truth, such as measuring the entropy of LLM responses, or neural NLI models trained on synthetic data, introducing biases. This carries the risk of introducing and propagating biases within the detection method, leading to a less robust solution.
3. **Lack of explainability**: Neural methods alone are not transparent, making it difficult to understand why certain outputs are classified as hallucinations, degrading user trust.

Method

Our proposed method (Figure 1) systematically detects and explains hallucinations in LLM outputs through comparison of KG representations. Initially, an LLM is employed for triple extraction on the text to obtain a KG representing its claims. For the open-domain problem, this claim KG is then paired with a ground truth KG derived from relevant triples within Wikidata, with both graphs capturing the relationships and attributes of the identified entities. For the closed-domain problem, the claim KG is compared with a KG constructed via the same LLM-based method applied to the provided context.

The KGs are then compared using a Weisfeiler-Lehman subtree kernel (Shervashidze et al. 2011), which provides a numerical measure of the structural similarity between the graph pair. In cases where the similarity score falls below a predetermined threshold, indicating a potential hallucination, an explanation is generated via a two-step analysis process. First, contradictory relations between the two KGs are identified based on embeddings of the triple entities. A modified graph edit distance algorithm is then used to identify the specific structural differences between the claim and ground-truth KGs. These differences are provided to an LLM to generate a contrastive explanation

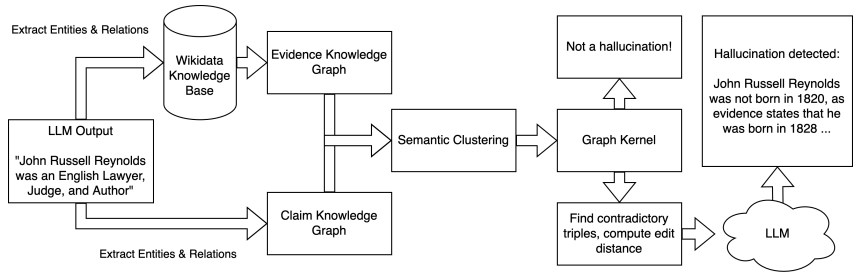


Figure 1. Visual depiction of the open-domain proposed method.

that explains why the original output qualifies as a hallucination, highlighting specific discrepancies between the model's claims and the established facts (Miller 2019).

Testing is conducted using publicly-available datasets. For the open-domain problem, testing is done on the WikiBio GPT-3 Hallucination Dataset (Manakul et al. 2023). This dataset consists of 238 long textual passages generated by GPT-3. Each passage is broken down into multiple parts, with each part being labeled as either "accurate", "minor inaccurate", or "major inaccurate". This dataset provides enough data and granular labels to produce a detailed analysis of the performance of the proposed method, and is widely used by existing open-domain hallucination detection approaches, allowing for ease of comparison. For the closed-domain problem, testing is done on the SummEval (Fabbri et al. 2021) and QAGS-C (Wang et al. 2020) datasets, which provide LLM summaries of given texts, making these datasets useful for evaluating hallucinations based on a source text.

Knowledge Graph Construction

The process for converting unstructured textual data into a KG is split into three key stages (Sansford et al. 2024):

1. **Named Entity Recognition (NER)**: Identifying atomic entities from the text, such as people, organizations, locations, dates, etc.;

2. **Coreference resolution**: Finding all mentions in the text that refer to the same entity and relabel the detected entities accordingly; and

3. **Relation extraction**: Identifying relations between the detected entities.

Large language models (LLMs) have become a standard tool for constructing KGs from text due to their ability to extract contextual features from text (Kommineni et al. 2024). AutoKG (Chen and Bertozzi 2023) demonstrates their potential for automated KG generation. Methods like GraphEval (Sansford et al. 2024) enhance performance via in-context learning and chain-of-thought (CoT) prompting to guide entity and relation extraction. Building on this, our approach applies similar prompting strategies to constrain the LLM to task-specific knowledge and reduce hallucinated triples. We

set temperature to 0 for fully deterministic responses and support modular upgrades to newer LLMs without modifying the core methodology. The full prompt is provided in the Appendix, under "LLM Prompts".

To evaluate LLM-generated KGs in open-domain tasks, we first retrieve ground-truth data from Wikidata, an extensive knowledge base of 114.7 million items, developed to serve as a central repository for structured data (Vrandečić and Krötzsch 2014). Advantages of using Wikidata here are the large coverage and linkage from each entity to a reference and qualifiers (providing additional context). Using the Spacy Entity Linker (Gerber 2024), detected entities are aligned with corresponding Wikidata entries, enabling SPARQL queries to extract relational triples between linked entities. These triples form the ground-truth KG for assessing the correctness and completeness of the LLM-generated KG. To enrich semantic depth, we also retrieve entity descriptions from Wikidata and Wikipedia, which are parsed using the same LLM-based method used to construct the claim KG. This added context improves coverage and strengthens comparison robustness.

Knowledge Graph Relation Selection

Because the context or ground-truth KG often contains a significant amount of information which is irrelevant to the claim KG, relation selection must be done to refine it so that the two KGs can be compared using the graph kernel. For this task, our method uses embeddings obtained from Sentence-BERT (SBERT), a popular BERT (Bidirectional Encoder Representations from Transformers) architecture designed specifically for tasks involving sentence similarity and semantic search (Reimers and Gurevych 2019). SBERT provides sentence embeddings that can be compared using cosine similarity. To obtain embeddings for KG triples, we concatenate the components into a string, e.g., the triple ("Albert Einstein", "was born in", "Ulm") is transformed into "Albert Einstein was born in Ulm". This concatenated string is then passed through SBERT to generate an embedding for the triple.

To select the most relevant triples from the context/ground-truth KG, we match each triple in the claim KG with the most semantically similar triple from the context/ground-truth KG. Relevance is determined by maximizing the cosine similarity between the embeddings of the current claim KG triple and a triple from the context/ground-truth KG. Specifically, the relevant triple $T_{context}$ from the context KG is identified as the one that maximizes the cosine similarity between its embedding $E_{context}$ and the embedding of the current claim KG triple T_{claim}:

$$T_{context}(T_{claim}) = \arg\max\left(\frac{E_{context} \cdot E_{claim}}{|E_{context}| \cdot |E_{claim}|}\right) \tag{1}$$

This process results in a filtered ground-truth KG, of size at most equal to the size of the claim KG. The ground-truth KG now only contains the most relevant triples to the claim.

Semantic Clustering of Knowledge Graph Labels

Since graph kernels have no knowledge of the semantic meaning of graph labels, a mechanism for dealing with semantically similar but syntactically differing labels between graphs is required. To achieve semantic comparison of labels, SBERT is used again to produce word embeddings for each node and edge label across both KGs. These embeddings are then clustered using an agglomerative hierarchical clustering algorithm. Cosine similarity is used as the cluster metric, with average linkage as the criterion (Ackermann et al. 2014). A distance threshold of 0.35 was chosen empirically, based on maximization of performance on hand-created tests as well as benchmarks covered in the Experiments section. The result allows for semantically similar labels to be clustered together under the same label, e.g., 'capital of France' and 'Paris' would both be grouped into the same cluster, and represented under the same label. This allows for our approach to deal with synonyms and contextually-similar phrases, such as 'capital of France' and 'Paris', while using the symbolic graph kernel method which has no knowledge of semanticity of labels.

Graph Kernel-based Comparison of Knowledge Graphs

Once the pre-processing steps have been applied to the KGs, the next step is to compute a comparison score between the KG pair using the Weisfeiler-Lehman (WL) subtree kernel (Shervashidze et al. 2011). This graph kernel is particularly useful in measuring the structural similarity between pairs of graphs, and it is well-suited for KGs that represent complex relationships between entities.

In general, a graph kernel is a function

$$k(G_1, G_2) = \langle \phi(G_1), \phi(G_2) \rangle,$$

where G_1, G_2 are graphs, and $\phi : \mathcal{G} \to \mathbb{R}^d$ is a feature map embedding each graph into a high-dimensional vector space. This formulation allows structural similarity to be quantified through an inner product.

The WL kernel achieves this by iteratively refining node labels through neighborhood aggregation. Initially, each node in the graph is labeled with its own numerical identity or a basic feature. Then, at each iteration, node labels are updated by concatenating their current label with those of their neighbors and hashing the result to form a new compressed label. The distribution of these refined labels across all iterations defines the feature vector $\phi(G)$. The kernel value is then computed as the inner product between the label count vectors of two graphs.

The WL kernel is effective for comparing KGs because it can capture graph isomorphisms and structural patterns at different levels of abstraction (such as at the entity level, relation level, and subtree level), while taking into account the presence of varying node labels or relational differences. Thus, we can compare the KGs in a manner that accounts for both the topology of the graphs and the semantic relationships (labels) encoded within them. This method is particularly effective in identifying subtle differences between KGs and for handling missing data, which is important in the context of detecting hallucinations or inconsistencies in KG-based representations of text.

Additionally, the WL kernel is efficient to run, with a time-complexity of $\mathcal{O}(hm)$, where h is the number of iterations and m is the total number of edges across both graphs.

Our implementation leveraged the open-source graph kernel library, GraKeL (Siglidis et al. 2020). This library enables the integration of kernel-based graph comparison into our workflow without significant time-investment in implementation of the kernel. For our method, we set the number of iterations to five and normalize the output of the kernel to return a similarity score between zero and one.

For a concrete worked example relevant to our method, see the Appendix, under "Weisfeiler-Lehman Graph Kernel Worked Example".

Generation of Explanations

To generate contrastive explanations, we first identify pairs of contradictory relations across the two original KGs (prior to relation selection). These are pairs of triples where two of the three elements are semantically similar, but the third differs significantly. We determine similarity using word embeddings and thresholding on cosine similarity. For instance, ("France", "capital city", "Paris") and ("France", "capital", "Rome") form a contradictory pair, as the first two elements align semantically, while the third does not. This approach identifies key discrepancies between the graphs, rather than simply unrelated triples. Next, a simplified graph edit distance algorithm (Algorithm 1) determines the sequence of edge operations required to transform the claim KG's contradictory relations into those of the ground-truth KG. In the above example, the algorithm identifies the need to add "Paris" as France's capital and remove "Rome".

We treat two triples (h_1, r_1, t_1) and (h_2, r_2, t_2) as semantically equivalent when the cosine similarity of *all* relations fall above a pre-set threshold.

Algorithm 1 Get Edit Operations Between Two KGs

Require: Two sets of triples T_1, T_2; thresholds τ_e, τ_r
function SEMEQ($(h_1, r_1, t_1), (h_2, r_2, t_2)$)
    return $\cos(h_1, h_2) \geq \tau_e \wedge \cos(r_1, r_2) \geq \tau_r \wedge \cos(t_1, t_2) \geq \tau_e$
end function
Initialise an empty list $operations$
for all $x \in T_1$ **do**
    if $\neg\exists y \in T_2 :$ SEMEQ(x, y) **then**
        Append (DELETE, x) to $operations$
    end if
end for
for all $y \in T_2$ **do**
    if $\neg\exists x \in T_1 :$ SEMEQ(x, y) **then**
        Append (ADD, y) to $operations$
    end if
end for
return $operations$

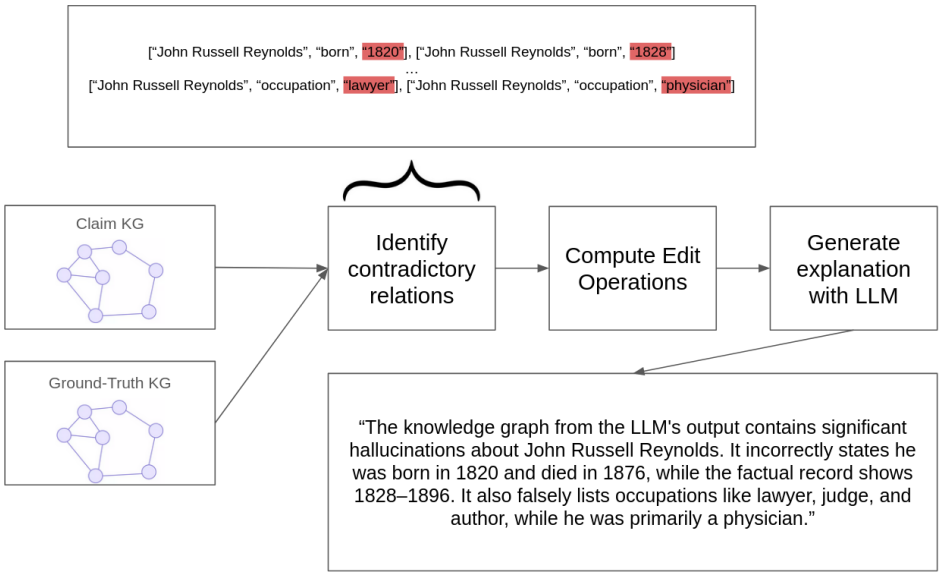


Figure 2. Explanation generation from the claim and ground-truth knowledge graphs.

Finally, an LLM (gpt-4o-mini) generates a natural language explanation of the detected hallucination based on these contradictory relations and edit operations. Unlike template-based methods, the LLM provides more flexible, natural, and detailed explanations. This enhances interpretability for human readers and helps to pinpoint the hallucinations. An example of this process is shown in Figure 2.

Experiments

For evaluation of our method, we carried out three separate experiments. The first of which focuses on the closed-domain hallucination detection problem, where the detection method has access to a specific, constrained context which is supplied alongside the LLM's generated content. The second experiment focuses on detection of open-domain hallucinations, where the detection method has access to external sources, such as a knowledge base as used in our method. Thirdly, we run an experiment to evaluate the efficacy of our method's generation of explanations for detected hallucinations.

In reporting the results of these experiments, we align with benchmark methods by using the same metrics for comparison. A comprehensive report of the results for all metrics (accuracy, balanced accuracy, precision, recall, and F1) is provided in the Appendix, under "Full Hallucination Detection Experiment Results". Source code for this project can be viewed at `https://github.com/Reih02/hallucination_explanation_graph_kernel_analysis`.

Benchmark	No. of Examples	Label Ratio	Avg length
SummEval	1,600	33.2%	63
QAGS-C	235	48.1%	49

Table 1. Statistics for the two evaluation benchmarks used. The label ratio is the ratio of consistent examples to hallucinatory examples.

Closed-Domain Hallucination Detection

Our evaluation approach employs two widely-used benchmarks for assessing closed-domain hallucination detection in the current literature. We compare our results to GraphEval's, a recent similar approach for hallucination detection that also employs KGs (Sansford et al. 2024). GraphEval's method uses these KGs to enhance existing Natural Language Inference (NLI) models. The first benchmark reported by GraphEval is SummEval (Fabbri et al. 2021), which evaluates summaries of news articles generated by language models using human ratings. The dataset incorporates 16 different language model outputs for each of 100 articles sourced from CNN and DailyMail, producing a dataset of 1600 total summaries. Evaluators rate these summaries from 1 to 5 across four dimensions: consistency, coherence, fluency, and relevance. We classify summaries with a consistency score lower than 4 as a hallucination, and higher than or equal to 4 as being consistent.

The second benchmark used by GraphEval is the QAGS-C dataset (Wang et al. 2020), derived from 235 articles, again obtained from CNN/DailyMail sources. The evaluation process for this benchmark involved human annotators conducting a sentence-level analysis of each summary, evaluating factual consistency by comparing individual sentences against the provided source article, and outputting a binary consistency rating. Each sentence underwent review by three separate annotators, with the final score made by taking the average of these three votes. In our analysis, we consider a sentence to be hallucinatory if it has an average consistency label less than 0.6, and higher than or equal to 0.6 indicates a consistent sentence. Detailed statistics for these two benchmarks can be seen in Table 1

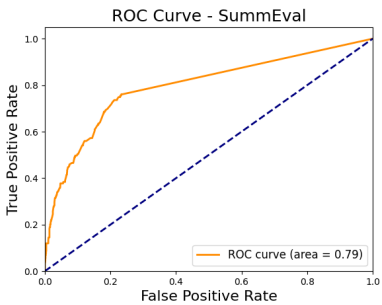
We empirically set graph kernel similarity thresholds at 0.15 (SummEval) and 0.5 (QAGS-C) for optimal balanced accuracy. Results (Table 2) show that our method outperforms four of six GraphEval methods on average, ranking just behind TrueTeacher, which benefits from synthetic data in fine-tuning, a bias risk we avoid due to the criticality of using a ground-truth source in hallucination detection. KEA Explain performs best on the SummEval task, with a balanced accuracy of 0.761 (second out of all methods). The performance in the QAGS-C task is slightly worse, coming in fourth overall, behind the three knowledge-graph enhanced methods. Overall, this shows that the performance of our method is comparable to existing closed-domain knowledge-graph based hallucination detection methods.

The ROC curves (Figure 3) visualize the performance across different graph kernel thresholds, with AUC values: 0.79 (SummEval) and 0.70 (QAGS-C). Both curves bend

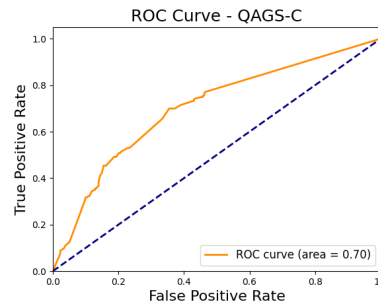
Method	SummEval	QAGS-C	Average Score
HHEM (He et al. 2021)	0.660	0.635	0.648
GraphEval (HHEM)	0.715	0.722	0.714
TRUE (Honovich et al. 2022)	0.613	0.618	0.616
GraphEval (TRUE)	0.724	0.717	0.721
TrueTeacher (Gekhman et al. 2023)	0.749	<u>0.756</u>	<u>0.753</u>
GraphEval (TrueTeacher)	**0.792**	**0.781**	**0.787**
KEA Explain (Ours)	<u>0.761</u>	0.711	0.736

Table 2. Comparison of balanced accuracy results for the SummEval and QAGS-C experiments. Bold indicates top-ranked, while underlined indicates second-ranked

Figure 3. ROC Curves for our classifier on different benchmarks.



(a) SummEval ROC



(b) QAGS-C ROC

toward the top-left, indicating high True Positive Rates (TPR) with low False Positive Rates (FPR). In SummEval's ROC curve, a linear segment at higher thresholds suggests the classifier approaches random guessing beyond a certain threshold.

There is also a significant difference in performance between the two benchmarks, with our method achieving noticeably better results on SummEval than on QAGS-C. We hypothesize that this difference arises from the shorter average sentence length in the QAGS-C benchmark compared with SummEval. These shorter sentences may lead to less informative knowledge graphs, making the graph kernel more susceptible to small discrepancies in knowledge representation between the LLM's summarization and the context that are not accounted for by the label clustering process.

Open-Domain Hallucination Detection

To assess our method's effectiveness in addressing open-domain hallucination detection, we utilized the WikiBio GPT-3 hallucination dataset. This dataset contains 238 GPT-3 (`text-davinci-003`)-generated passages that resemble Wikipedia-style content.

Method	Precision	Recall	F1
SelfCheckGPT	**0.843**	0.917	0.879
AlignScore	0.809	0.981	**0.886**
KEA Explain (Ours)	0.734	**0.984**	0.841

Table 3. Comparison of Precision, Recall, and F1 results for the WikiBio dataset.

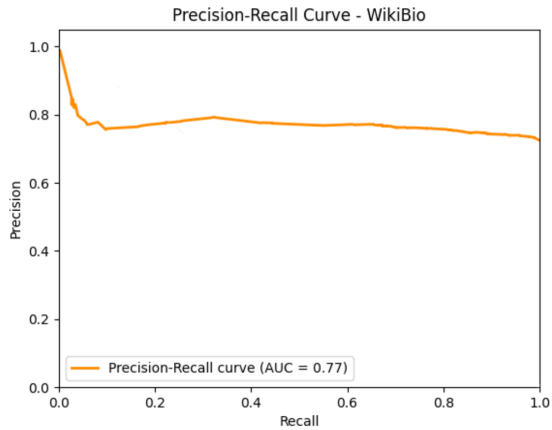


Figure 4. Precision–Recall Curve on the WikiBio benchmark.

Each passage is segmented into sentences and annotated as being either accurate, containing minor inaccuracies, or containing major inaccuracies. Sentences labeled with minor or major inaccuracies are considered hallucinatory, while accurate sentences are classified as consistent. This dataset is unbalanced, with just 27.0% of the 1,908 sentences being consistent.

We report comparisons with SelfCheckGPT (Manakul et al. 2023) and AlignScore (Zha et al. 2023), two open-domain approaches, using the results reported by the authors of FactAlign (Rashad et al. 2024), a closed-domain method. We follow these papers by reporting Precision, Recall, and F1 scores, optimizing the graph kernel similarity threshold to 0.3 in order to maximize the F1 score. Our method achieved a similar F1 score to the other methods but had lower precision, indicating a tendency to classify non-hallucinated responses as hallucinations (Table 3). Despite this, our method outperformed the other benchmarks in recall suggesting that it is particularly effective at detecting true positives (hallucinations) in open-domain settings.

Based on manual inspection of failed classifications, we hypothesize that the lower precision stems from limitations in the knowledge base querying process, particularly the inability to retrieve niche or highly-specific entities from Wikidata, increasing the likelihood of false positives. While this reliance on Wikidata leads to more false positives, it also provides a ground-truth reference for factual accuracy, which other methods lack

(Manakul et al. 2023; Zha et al. 2023). Both SelfCheckGPT and AlignScore also showed lower precision compared to recall, indicating that handling of false positive predictions is a common challenge in open-domain hallucination detection.

Due to the class imbalance in the WikiBio dataset, we report a Precision-Recall curve (Figure 4). We observe an AUC of 0.77 for this PR curve. The curve shows an initial sharp precision drop as recall increases from 0 to 0.1 when the graph kernel threshold decreases. Precision then stabilizes around 0.75-0.8 across most recall values (0.1 to 0.9). With 73% of examples being hallucinatory (our positive class), this majority helps maintain relatively high precision throughout the curve. As thresholds decrease further, the model captures more true hallucinations while precision remains steady because correctly identifying the abundant positive class offsets the increasing false positives. The AUC of 0.77 reflects this balanced performance.

Hallucination Explanations

To evaluate the effectiveness of our method for generating explanations of detected hallucinations, we adopt a set of criteria for assessing the "goodness" of an explanation, inspired by the "Explanation Goodness Checklist" proposed by (Johs 2024) (see Table 4 for qualitative rating descriptions of the four criteria). For our evaluation, we randomly selected 20 entries with a consistency label ≤ 3 from the SummEval dataset (see Appendix, under "Example Explanations," for examples of generated explanations). Each explanation was evaluated against the corresponding original article and generated summary using these criteria to determine how well it captured the hallucination(s) present. We focus on the closed-domain setting for this evaluation, as it allows for more straightforward manual assessment using the provided ground truth.

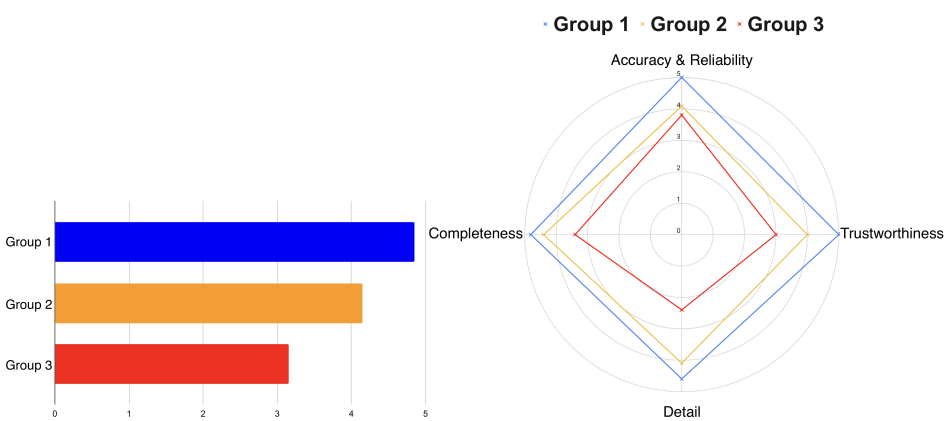
We defined three groups based on the consistency rating label of the LLM output for which the explanation was generated: Group 1 contains consistency ratings between 0 and 1; Group 2 ratings between 1 and 2; and Group 3 ratings between 2 and 3, such that Group 1 contains the most severe hallucinations, and Group 3 the least.

In Group 1, the average score based on the rating criteria was 4.85, while Group 2 received 4.15, and Group 3 scored 3.15. These results reveal a clear trend: as the consistency of the output increases (indicating a less-significant hallucination), explanation quality declined. This monotonic pattern held across all four evaluation criteria (Figure 5). Notably, while the accuracy and reliability of the explanations remained relatively close across the groups, the most significant decline in ratings was observed in the "Level of Detail" criterion. This suggests that, as the significance of the hallucination in the LLM's output decreases, more details are omitted from the explanation that could have enhanced its quality. We hypothesize that this effect is due to the decreased likelihood of finding relevant conflicting triples to guide the explanation generation process when the hallucinations are more sparse and nuanced. In this case, the explanation-generation process has less guidance as to exactly where the hallucination occurred, and crucial details could get left out of the generated explanation.

Criteria	Rating descriptions
Level of Detail	1: Too brief, lacks necessary detail. 2: Somewhat detailed 3: Adequate detail, could be deeper. 4: Well-detailed, covers most aspects. 5: Comprehensive, in-depth.
Completeness of Explanation	1: Incomplete, misses major aspects. 2: Some key components missing. 3: Mostly complete, with minor gaps. 4: Almost complete. 5: Fully complete, covers all aspects.
Accuracy and Reliability	1: Inaccurate, major errors. 2: Some inaccuracy or missing context. 3: Mostly accurate, small errors. 4: Very accurate, minor issues. 5: Fully accurate and reliable.
Trustworthiness	1: No supporting evidence. 2: Lacking sufficient evidence. 3: May lack evidence. 4: Well-supported claims. 5: Trustworthy, solid evidence.

Table 4. Criteria for rating generated explanations.

Figure 5. Average ratings across all four features.



(a) For each group.

(b) Across each consistency range.

Discussion

Key Findings and Advantages

In our closed-domain hallucination detection experiment, our method achieved a balanced accuracy of 0.736, outperforming four of the six methods reported in GraphEval

(Sansford et al. 2024) and exceeding their average score of 0.707. We also observed here that our method seemed to achieve a significantly higher score on the SummEval benchmark compared to the QAGS-C benchmark, which we hypothesize is due to the longer sentences supplied in the SummEval benchmark. Longer sentences yield denser and more informative knowledge graphs, which our graph kernel approach can exploit more effectively. By considering the surrounding subgraph structure rather than relying on individual triples, the method remains robust when some information is missing or noisy—an advantage over triple-matching methods such as FactAlign (Rashad et al. 2024) and GraphEval (Sansford et al. 2024). Shorter sentences tend to provide less informative knowledge graphs, which can make the graph kernel similarity calculation more sensitive to minor discrepancies, potentially leading to false predictions.

For the open-domain tests, we compared our method to two benchmarked approaches, SelfCheckGPT (Manakul et al. 2023) and AlignScore (Zha et al. 2023), reporting precision, recall, and F1. Our method achieved a lower precision (0.734) but a higher recall (0.984) than both baselines, resulting in an F1 score of 0.841. This strong recall can be explained by the reliance on Wikidata as a structured factual source: niche or highly specific entities may be missed (lowering precision), but the use of entity linking to a broad knowledge base ensures that hallucinations are rarely overlooked. This is comparable to the state of the art and supports our claim of similar performance in hallucination detection. The high recall and lower precision suggest that our method tends to falsely label some non-hallucinatory examples as hallucinations. We attribute this to its reliance on entity matching against the Wikidata knowledge base, as niche or highly specific entities are less likely to have matches, increasing the chance of false positives. Conversely, using Wikidata as a ground-truth source does enable our method to outperform others in recall, making it the best at correctly identifying hallucinations when they occur. This reflects a trade-off inherent in ground-truth methods that require entity matching.

Although we have shown performance close to state-of-the-art detection methods, our key contribution is represented in our method's ability to add interpretability through the use of graph-based knowledge representation. Here, KEA Explain's neurosymbolic design is crucial: by combining symbolic techniques (KGs, graph kernels, clustering) with neural techniques (word embeddings, transformers), the system not only achieves competitive detection performance but also produces explanations. These explanations provide contrastive evidence of why and where a hallucination occurred, something not offered by LLM-polling or strictly neural methods such as AlignScore, ChainPoll (Friel and Sanyal 2023), or SelfCheckGPT. In this experiment, we observed a monotonic decrease in explanation quality ratings across all four evaluation features (completeness, accuracy & reliability, trustworthiness, and detail) as consistency of the explanations increased. This suggests an inherent trade-off: while severe hallucinations are easier to explain with detailed conflicting triples, subtler ones provide fewer cues, reducing explanation richness.

Across both closed-domain and open-domain evaluations, these findings show that our method achieves results comparable to state-of-the-art approaches, while additionally offering a degree of interpretability that most other methods lack. The advantages of

structural comparison through graph kernels and neurosymbolic integration directly account for the method's robustness and explanatory power.

Limitations and Future Research

Despite its strong results, our method has some limitations that point toward valuable directions for future research.

The first limitation of our proposed method lies in the graph kernel similarity calculation. As observed in our experiments, different domains and tasks can lead to differing optimal graph kernel similarity thresholds being used. For example, we arrived at 0.15 for SummEval, 0.50 for QAGS-C, and 0.30 for WikiBio. This sensitivity to different domains and tasks adds complexity for real-world implementation, as it may be required to optimize the threshold before use. A practical solution is to develop adaptive or automated threshold selection methods, for example by leveraging labeled datasets from the target domain or by exploring unsupervised approaches that learn thresholds dynamically.

The second limitation concerns the method's reliance on a knowledge base such as Wikidata, which can increase false-positive predictions. This occurs when the entity-linking process fails to accurately identify highly specific or obscure entities that are poorly represented in the knowledge base. In such cases, entities with no direct match may be incorrectly linked to more general or irrelevant entries, leading to inaccurate predictions. This limitation is an inherent trade-off of relying on an external ground-truth source: while it enables predictions based on structured data and improves the true positive rate, it also raises the risk of false positives for underrepresented entities. Despite this, Wikidata and similar resources remain valuable overall, as they provide a strong factual basis for hallucination detection. Moreover, it is generally more important for hallucination detection systems to maintain strong recall than high precision, as false negatives (i.e., hallucinations that are predicted as being factually correct) can undermine trust in the system. Future work could mitigate this limitation by broadening the ground-truth sources—for instance, integrating multiple knowledge bases or supplementing them with large-scale web corpora to increase coverage and improve precision. Such an approach would directly address the current trade-off between strong recall and lower precision.

Third, the requirements of having to construct ground truth KGs can induce a significant computational burden, which may make our method less effective in time-constrained applications and domains. This is especially prevalent in the open-domain, where many relevant entities must first be retrieved from Wikidata via SPARQL queries.

Finally, our approach to explanation generation has its own limitations. The method, which relies on a simplified graph edit distance algorithm, contradictory-triple identification, and an LLM, showed limitations when hallucinations were more subtle (i.e., when consistency was higher). Future research could explore alternative explanation and correction strategies, such as through generation of follow-up queries to the LLM that generated the hallucinated sentence, which would further enhance the method's

transparency and practical utility, as well as have the potential to correct hallucinated facts.

Acknowledgements

This work is part of the "Beyond Prediction: explanatory and transparent data science" project supported by the Strategic Science Investment Fund administered by the Ministry of Business Innovation and Employment, Aotearoa/New Zealand.

We also wish to thank the editors of this work for their valuable comments and suggestions, which helped improve the clarity and quality of this paper.

References

Ackermann MR, Blömer J, Kuntze D and Sohler C (2014) Analysis of agglomerative clustering. *Algorithmica* 69: 184–215.

Almeida F and Xexéo G (2019) Word embeddings: A survey. *arXiv preprint arXiv:1901.09069* .

Arrieta AB, Díaz-Rodríguez N, Del Ser J, Bennetot A, Tabik S, Barbado A, García S, Gil-López S, Molina D, Benjamins R et al. (2020) Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion* 58: 82–115.

Arslan M, Ghanem H, Munawar S and Cruz C (2024) A survey on rag with llms. *Procedia computer science* 246: 3781–3790.

Chen B and Bertozzi AL (2023) Autokg: Efficient automated knowledge graph generation for language models. In: *2023 IEEE International Conference on Big Data (BigData)*. IEEE, pp. 3117–3126.

Chen D and Yih Wt (2020) Open-domain question answering. In: *Proceedings of the 58th annual meeting of the association for computational linguistics: tutorial abstracts*. pp. 34–37.

Chen X, Jia S and Xiang Y (2020) A review: Knowledge reasoning over knowledge graph. *Expert systems with applications* 141: 112948.

Dwivedi R, Dave D, Naik H, Singhal S, Omer R, Patel P, Qian B, Wen Z, Shah T, Morgan G et al. (2023) Explainable ai (xai): Core ideas, techniques, and solutions. *ACM Computing Surveys* 55(9): 1–33.

Fabbri AR, Kryściński W, McCann B, Xiong C, Socher R and Radev D (2021) Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics* 9: 391–409.

Friel R and Sanyal A (2023) Chainpoll: A high efficacy method for llm hallucination detection. *arXiv preprint arXiv:2310.18344* .

Gao Y, Xiong Y, Gao X, Jia K, Pan J, Bi Y, Dai Y, Sun J, Wang M and Wang H (2023) Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* .

Gekhman Z, Herzig J, Aharoni R, Elkind C and Szpektor I (2023) TrueTeacher: Learning factual consistency evaluation with large language models. In: Bouamor H, Pino J and Bali K (eds.)

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Singapore: Association for Computational Linguistics, pp. 2053–2070.

Gerber E (2024) Spacy entity linker. https://github.com/egerber/spacy-entity-linker.

Guan X, Liu Y, Lin H, Lu Y, He B, Han X and Sun L (2024) Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38. pp. 18126–18134.

He P, Liu X, Gao J and Chen W (2021) DeBERTa: Decoding-enhanced BERT with disentangled attention. In: *International Conference on Learning Representations (ICLR)*. URL https://openreview.net/forum?id=XPZIaotutsD.

Hogan A, Blomqvist E, Cochez M, d'Amato C, Melo GD, Gutierrez C, Kirrane S, Gayo JEL, Navigli R, Neumaier S et al. (2021) Knowledge graphs. *ACM Computing Surveys (Csur)* 54(4): 1–37.

Honovich O, Aharoni R, Herzig J, Taitelbaum H, Kukliansy D, Cohen V, Scialom T, Szpektor I, Hassidim A and Matias Y (2022) TRUE: Re-evaluating factual consistency evaluation. In: Carpuat M, de Marneffe MC and Meza Ruiz IV (eds.) *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, pp. 3905–3920.

Huang L, Yu W, Ma W, Zhong W, Feng Z, Wang H, Chen Q, Peng W, Feng X, Qin B et al. (2025) A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems* 43(2): 1–55.

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

Johs AJ (2024) *A Qualitative Investigation of Explanation Goodness in a Biomedical AI System*. PhD Thesis, Drexel University.

Kommineni VK, König-Ries B and Samuel S (2024) From human experts to machines: An llm supported approach to ontology and knowledge graph construction. *arXiv preprint arXiv:2403.08345* .

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*. pp. 9004–9017.

Miller T (2019) Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267: 1–38.

Murtagh F and Contreras P (2012) Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2(1): 86–97.

Oelschlager R (2024) Evaluating the impact of hallucinations on user trust and satisfaction in LLM-based systems. https://www.diva-portal.org/smash/get/diva2:1870904/FULLTEXT01.pdf.

Perković G, Drobnjak A and Botički I (2024) Hallucinations in LLMs: Understanding and addressing challenges. In: *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*. pp. 2084–2088.

Rashad M, Zahran A, Amin A, Abdelaal A and Altantawy M (2024) FactAlign: Fact-level hallucination detection and classification through knowledge graph alignment. In: Ovalle A, Chang KW, Cao YT, Mehrabi N, Zhao J, Galstyan A, Dhamala J, Kumar A and Gupta R (eds.) *Proceedings of the 4th Workshop on Trustworthy Natural Language Processing (TrustNLP 2024)*. Mexico City, Mexico: Association for Computational Linguistics, pp. 79–84.

Reimers N and Gurevych I (2019) Sentence-bert: Sentence embeddings using siamese bert-networks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. pp. 3982–3992.

Sansford H, Richardson N, Maretic HP and Saada JN (2024) GraphEval: A knowledge-graph based LLM hallucination evaluation framework. In: *KiL@KDD*. URL https://ceur-ws.org/Vol-3894/paper5.pdf.

Shervashidze N, Schweitzer P, Van Leeuwen EJ, Mehlhorn K and Borgwardt KM (2011) Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research* 12(9).

Siglidis G, Nikolentzos G, Limnios S, Giatsidis C, Skianis K and Vazirgiannis M (2020) GraKeL: A graph kernel library in Python. *Journal of Machine Learning Research* 21(54): 1–5.

Tonmoy S, Zaman S, Jain V, Rani A, Rawte V, Chadha A and Das A (2024) A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313* 6.

Vishwanathan SVN, Schraudolph NN, Kondor R and Borgwardt KM (2010) Graph kernels. *The Journal of Machine Learning Research* 11: 1201–1242.

Vrandečić D and Krötzsch M (2014) Wikidata: a free collaborative knowledgebase. *Communications of the ACM* 57(10): 78–85.

Wang A, Cho K and Lewis M (2020) Asking and answering questions to evaluate the factual consistency of summaries. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. pp. 5008–5020.

Zha Y, Yang Y, Li R and Hu Z (2023) AlignScore: Evaluating factual consistency with a unified alignment function. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 11328–11348.

Definition of the Weisfeiler-Lehman Graph Kernel

A graph kernel is a function $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ that measures the similarity between two graphs, where $\mathcal{G}$ denotes the set of graphs. Graph kernels are a type of kernel function used in machine learning to enable the application of algorithms, such as Support Vector Machines (SVMs), to structured data like graphs. They compute the similarity between graphs by comparing their structural and attribute-based features Vishwanathan et al. (2010).

Mathematically, a graph kernel can be expressed as an inner product in a high-dimensional feature space:

$$k(G_1, G_2) = \langle \phi(G_1), \phi(G_2) \rangle \tag{2}$$

Where G_1 and G_2 are graphs, and $\phi : \mathcal{G} \to \mathbb{R}^d$ is a feature mapping that embeds graphs into a d-dimensional vector space. These kernels provide a way to quantify how 'similar' two graphs are by capturing structural patterns at different levels of abstraction, such as common subgraphs, paths, and tree structures present across the pair.

In this project, we utilize the Weisfeiler-Lehman Subtree Kernel Shervashidze et al. (2011), which leverages the Weisfeiler-Lehman graph isomorphism test to generate subtree patterns for comparison. To define the Weisfeiler-Lehman Subtree Kernel, let us assume the following:

- G, G' are two graphs
- $\Sigma_i \subseteq \Sigma$ are the set of letters that occur as node labels at least once in G or G' at the end of the i^{th} iteration of the Weisfeiler-Lehman algorithm
- Σ_0 is the set of original node labels of G and G'
- We define a map, $c_i : \{G, G'\} \times \Sigma_i \to \mathbb{N}$ such that $c_i(G, \sigma_{ij})$ is the number of occurrences of the letter σ_{ij} in the graph G.

The Weisfeiler-Lehman subtree kernel on two graphs G and G' with h iterations is then defined as the following:

$$k(G, G') = \langle \phi(G), \phi(G') \rangle \tag{3}$$

where

$$\phi(G) = (c_0(G, \sigma_{01}), ..., c_o(G, \sigma_{0|\Sigma_0|}), ..., c_h(G, \sigma_{h1}), ..., c_h(G, \sigma_{h|\Sigma_h|})) \tag{4}$$

and

$$\phi(G') = (c_0(G', \sigma_{01}), ..., c_o(G', \sigma_{0|\Sigma_0|}), ..., c_h(G', \sigma_{h1}), ..., c_h(G', \sigma_{h|\Sigma_h|})) \tag{5}$$

It can be shown that the Weisfeiler-Lehman Subtree Kernel can be computed in $\mathcal{O}(hm)$ time, where h is the number of iterations and m is the total number of edges across all graphs.

Weisfeiler-Lehman Graph Kernel Worked Example

Here, we illustrate how the Weisfeiler-Lehman (WL) kernel compares two small KGs using node-label counts across iterations, to serve as a concrete example relevant to our methodology.

Graphs.
G_1: (France, capital, Paris), (France, currency, Euro)
G_2: (France, capital, Paris), (France, currency, Franc)

We run the WL graph kernel for $h = 2$ iterations. Let Σ_i be the set of node labels after iteration i, and let $c_i(G, \sigma)$ count occurrences of label $\sigma \in \Sigma_i$ in G.

Iteration $i = 0$ (original labels).
Label multisets:
G_1: {`France`, `capital`, `Paris`, `currency`, `Euro`},
G_2: {`France`, `capital`, `Paris`, `currency`, `Franc`}.
Shared labels: 4 (`France`, `capital`, `Paris`, `currency`).

Iteration $i = 1$ (1-hop aggregation).
Each node is relabeled by hashing its current label with the multiset of its neighbors' labels. Nodes `capital` and `Paris` receive identical new labels in both graphs; `currency` differs because its neighbor is `Euro` versus `Franc`; `France` matches at i=1. Shared new labels: 3.

Iteration $i = 2$ (propagation).
Now `France` aggregates neighbors' *updated* labels (`capital`* and `currency`*). Since `currency`* already diverged at i=1, `France` also diverges at i=2; `capital` and `Paris` remain matched. Shared new labels: 2.

The WL feature map concatenates the label-count vectors across iterations, $\phi(G) = \big(c_0(G,\cdot), c_1(G,\cdot), \ldots, c_h(G,\cdot)\big)$, and the kernel is $k(G_1, G_2) = \langle \phi(G_1), \phi(G_2) \rangle$. With one occurrence per node label here, the cross-kernel sums the number of shared labels per iteration:

$$k(G_1, G_2) = 4\ (i{=}0)\ +\ 3\ (i{=}1)\ +\ 2\ (i{=}2)\ =\ 9.$$

The self-kernels are $k(G_1, G_1) = k(G_2, G_2) = 5 + 5 + 5 = 15$. A cosine-style normalization gives a similarity of $9/\sqrt{15 \cdot 15} = 0.6$. This quantifies how a local change (`Euro`→`Franc`) ripples one hop further at i=2, lowering the similarity as WL depth increases. Importantly, in our methodology, we set $i = 5$, which will have the effect of propagating local information further around the graph. This was a decision based on empirical performance, and should be adjusted depending on the domain and average graph size.

Explanations

Example Explanations

Sample 1 The knowledge graph generated from the LLM's output contains several key inaccuracies that highlight its hallucinatory nature. Notably, it incorrectly asserts a relationship between 'Space Invaders' and the year '1970', suggesting that the game was developed in that specific year. However, this claim is not supported by factual data, as the actual development period is more accurately described as occurring in the late 1970s. This discrepancy indicates a significant misunderstanding or misrepresentation of the timeline associated with the game's creation. Furthermore, the absence of the correct relationship linking 'Space Invaders' to the late 1970s in the LLM's output further emphasizes the inaccuracies present in the generated knowledge graph. Together, these false claims illustrate how the LLM's output diverges from established facts, leading to a misleading representation of the game's historical context.

Sample 2 The knowledge graph generated from the LLM's output contains several significant hallucinations that misrepresent the facts surrounding Ben Stokes and his experiences in cricket. Firstly, the LLM incorrectly states that Stokes "broke his neck" during the Ashes series, which is a serious misrepresentation; in reality, he broke his wrist after punching a locker, an incident that occurred the previous year. This error not only alters the nature of the injury but also misplaces the context of his struggles, as the original text emphasizes his need to manage his aggression rather than suggesting a severe injury like a broken neck. Additionally, the LLM's output inaccurately implies that Stokes is currently at the Kensington Oval, when in fact, the context suggests he is back in the England team and preparing for a match in Barbados. The relationship between Kensington Oval and England is also misrepresented, as the original context indicates a more nuanced connection, specifically that the Oval is a venue where Stokes has faced challenges. These inaccuracies collectively distort the narrative of Stokes's character and his journey, leading to a misleading portrayal of his situation in the England cricket team.

Full Hallucination Detection Experiment Results

Benchmark	Accuracy	Balanced Accuracy	Precision	Recall	F1
SummEval	0.782	**0.761**	0.276	0.736	0.401
QAGS-C	0.738	**0.711**	0.492	0.656	0.562
WikiBio	0.730	0.523	0.734	0.984	**0.841**

Table 5. Full hallucination-detection experiment results of our method across each benchmark. Metrics are marked in bold where they were the main focus of the benchmark comparison, and hence were optimised for in the graph kernel threshold selection process.

LLM Prompts

Knowledge Graph Construction

The following is the detailed prompt used for generating knowledge graphs from unstructured text. It was heavily inspired by Sansford et al. (2024):

```
messages=[{"role": "system", "content": "You are an expert at
    creating knowledge graphs based on text.\n"
     "You will receive two separate pieces of text, and you must
    perform the following steps on each piece of text:\n"
     "1. Entity detection: Select key and crucial entities from
    the text. Keep these entities short and concise and skip less
     important details of the text\n"
     "2. Coreference resolution: Across both texts, ensure that
    you use the same entity name for the same concept. For
    example, \"He\" may actually refer to the entity \"Peter\".
    Also apply this step between texts, so that the two knowledge
     graphs can be compared as easily as possible without
    confusion.\n"
     "3. Relation extraction: Identify semantic relationships
    between detected entities. These relationships should be
    encapsulated as a simple and concise relation such as \"began
     in\", or \"will simulcast\", for example.\n"
     "4. Knowledge Graph refinement: Once the two knowledge
    graphs have been created, try to ensure that similar triples
    between the two texts / knowledge graphs are represented the
    same way, to avoid confusion. For example, if two different
    entities refer to a similar event or concept, relabel them to
     be the same across the two knowledge graphs.\n\n"
     "Format your response as a JSON object that can be directly
    parsed without any edits to your response. This means that
    you are not allowed to include any text not part of the
    knowledge graphs.\n"
     "In the JSON object, one element should be the knowledge
    graph for the first text, and another element should be the
    knowledge graph for the second text.\n"
     "Each knowledge graph should be a list of triples, with each
     triple being a python list of the form [\"Peter\", \"height
    \", \"180cm\"]\n\n"
     "See below for some examples:\n\n"
     f"TEXT1: \n{sample_text1}\n\nTEXT2:\n{sample_text2}\n\n"
     "YOUR OUTPUT:\n"
     "{\n"
     "  \"knowledge_graph1\": [\n"
     "      [\"A&E Networks\", \"will simulcast in 2016\", \"
    Roots\"],\n"
     "      [\"Roots\", \"premiered in\", \"1977\"],\n"
     "      [\"Roots\", \"ran for\", \"four seasons\"],\n"
     "      [\"Roots\", \"instance of\", \"miniseries\"],\n"
     "      [\"Roots\", \"followed\", \"Kunta Kinte\"],\n"
```

```
    "      [\"Kunta Kinte\", \"was sold into\", \"slavery\"],\n"
    "      [\"Kunta Kinte\", \"was a\", \"free black man\"]\n"
    "  ],\n"
    "  \"knowledge_graph2\": [\n"
    "      [\"Roots\", \"one of the\", \"biggest TV events of
  all time\"],\n"
    "      [\"Roots\", \"had a staggering audience of\", \"over
  100 million viewers\"],\n"
    "      [\"Roots\", \"being\", \"reimagined for new audiences
  \"],\n"
    "      [\"Roots\", \"was about\", \"an African-American
  slave and his descendants\"],\n"
    "      [\"Roots\", \"premiered\", \"1977\"]\n"
    "  ]\n"
    "}\n\n ..."
    {"role": "user", "content": f"TEXT1: \n{text1}\n\nTEXT2:\n{
  text2}"}]
```

Natural Language Explanation

Below is the full prompt used to instruct the LLM to generate natural language explanations for detected hallucinations, guided by the extracted graph edit distance operations.

```
messages = [
  {
    "role": "system",
    "content": (
      "Your task is to generate a contrastive explanation as to
  why a knowledge "
      "graph produced from an LLM's output contains an
  hallucination.\n"
      "To achieve this task, you will receive a list of
  explanations obtained from "
      "running a graph edit distance algorithm between the LLM's
  output knowledge graph "
      "and a ground truth knowledge graph.\n"
      "This list of explanations shows the steps needed to
  transform the LLM's output "
      "knowledge graph into the ground truth knowledge graph,
  effectively capturing the "
      "hallucinatory components of the LLM's output.\n"
      "Instead of listing off each hallucination, try to tie it
  together into a paragraph "
      "to discuss the key false claims that the LLM's output
  knowledge graph contains.\n"
      "You will also receive the LLM's output text, and the
  original context that the "
```

```
      "LLM used to generate the summary of. Use context and
    common sense from these "
      "three pieces of information to guide your explanation."
    )
  },
  {
    "role": "user",
    "content": f"###Context###:\n{context}\n\n###LLM summary/
    output###:\n{summary}\n\n###Explanations###:\n{explanations}"
  }
]
```