FactGenius: Combining Zero-Shot Prompting and Fuzzy Relation Mining to Improve Fact Verification with Knowledge Graphs

Sushant Gautam
SimulaMet & OsloMet
Oslo, Norway
sushant@simula.no

Roxana Pop
University of Oslo
Oslo, Norway
roxanap@ifi.uio.no

Abstract

Fact-checking is a crucial natural language processing (NLP) task that verifies the truthfulness of claims by considering reliable evidence. Traditional methods are labour-intensive, and most automatic approaches focus on using documents as evidence. In this paper, we focus on the relatively understudied fact-checking with Knowledge Graph data as evidence and experiment on the recently introduced FactKG benchmark. We present FactGenius, a novel method that enhances fact-checking by combining zero-shot prompting of large language models (LLMs) with fuzzy text matching on knowledge graphs (KGs). Our method employs LLMs for filtering relevant connections from the graph and validates these connections via distance-based matching. The evaluation of FactGenius on an existing benchmark demonstrates its effectiveness, as we show it significantly outperforms state-of-the-art methods. The code and materials are available at https://github.com/SushantGautam/FactGenius.

1 Introduction

Fact-checking is a critical task in natural language processing (NLP) that involves automatically verifying the truthfulness of a claim by considering evidence from reliable sources (Thorne et al., 2018). This task is essential for combating misinformation and ensuring the integrity of information in digital communication (Cotter et al., 2022). Traditional fact-checking is performed by domain experts and is a labour-intensive process. Automatic fact-checking systems have been introduced to address this, but most of them work with textual data as evidence sources (Vladika and Matthes, 2023).

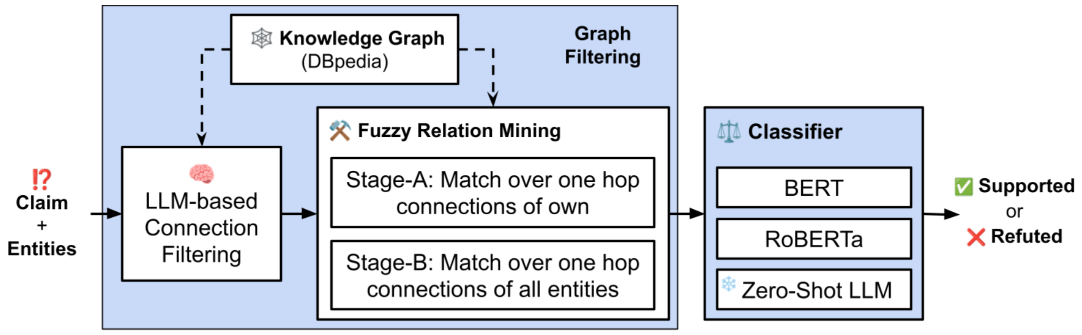
Recent advancements in large language models (LLMs) have shown promise in enhancing fact-checking capabilities (Choi and Ferrara, 2024). LLMs, with their extensive pre-training on diverse textual data, possess a vast amount of embedded knowledge (Yang et al., 2024). However, their outputs can sometimes be erroneous or lacking in specificity, especially when dealing with complex reasoning patterns required for fact-checking. External knowledge, such as knowledge graphs (KGs) (Hogan et al., 2021), can aid in fact-checking.

In this paper, we propose FactGenius, a novel approach that combines zero-shot prompting of LLMs with fuzzy relation-mining techniques to improve reasoning on knowledge graphs. Specifically, we leverage DBpedia (Lehmann et al., 2015), a structured source of linked data, to enhance the accuracy of fact-checking tasks.

Our methodology involves using the LLM to filter potential connections between entities in the KG, followed by refining these connections through Levenshtein distance-based fuzzy matching. This two-stage approach ensures that only valid and relevant connections are considered, thereby improving the accuracy of fact-checking.

We evaluate our method using the FactKG dataset (Kim et al., 2023b), which comprises 108,000 claims constructed through various reasoning patterns applied to facts from DBpedia. Our experiments demonstrate that FactGenius significantly outperforms existing baselines (Kim et al., 2023a), particularly when fine-tuning RoBERTa (Liu et al., 2019) as a classifier, achieving superior performance across different reasoning types.

In summary, the integration of LLMs with KGs and the application of fuzzy matching techniques represent a promising direction for advancing fact-checking methodologies. Our work contributes to this growing body of research by proposing a novel approach that effectively combines these elements, yielding significant improvements in fact-checking performance.



Knowledge Graph (DBpedia)
Graph Filtering
Claim + Entities
LLM-based Connection Filtering
Fuzzy Relation Mining
Stage-A: Match over one hop connections of own
Stage-B: Match over one hop connections of all entities
Classifier
BERT
RoBERTa
Zero-Shot LLM
Supported or Refuted

Figure 1: Overall pipeline of FactGenius: The process starts with LLM-based Connection Filtering using a knowledge graph (see Section 4.1.1). In Fuzzy Relation Mining (see Section 4.1.2), Stage-I matches one-hop connections of entities, and optionally, Stage-II includes all entities' connections. The classifier (BERT, RoBERTa, or Zero-Shot LLM; see Section 4.3) then determines if the claim is supported or refuted.

2 Literature Review

Fact-checking has become an increasingly vital aspect of natural language processing (NLP) due to the proliferation of misinformation in digital communication (Guo et al., 2022). Traditional approaches to fact-checking have typically relied on manually curated datasets and rule-based methods. While these methods are effective in controlled environments, they often struggle with scalability and adaptability to new types of misinformation (Saquete et al., 2020; Guo et al., 2022). The labor-intensive nature of these methods also poses significant challenges in rapidly evolving information landscapes (Nakov et al., 2021; Zeng et al., 2021).

To address challenges in understanding machine-readable concepts in text, FactKG introduces a new dataset for fact verification using claims, leveraging knowledge graphs (KGs) to encompass diverse reasoning types and linguistic patterns. This approach aims to enhance the reliability and practicality of KG-based fact verification (Kim et al., 2023b). Similarly, the Fact Extraction and VERification (FEVER) dataset (Thorne et al., 2018) pairs claims with Wikipedia sentences that support or refute them, providing a benchmark for fact-checking models. The authors employed a combination of natural language inference models and information retrieval systems to assess claim veracity.

The GEAR framework (Zhou et al., 2019) improves fact verification by using a graph-based method to aggregate and reason over multiple pieces of evidence. This approach surpasses previous methods by enabling more interactive and effective use of evidence.

Recent advancements in large language models (LLMs) have demonstrated considerable potential for enhancing fact-checking processes (Kim et al., 2023a; Choi and Ferrara, 2024). LLMs are pre-trained on vast and diverse corpora (Yang et al., 2024), allowing them to generate human-like text and possess a broad knowledge base (Choi and Ferrara, 2024). However, despite their impressive capabilities, LLMs can sometimes produce erroneous outputs or lack the specificity required for complex fact-checking tasks (Choi and Ferrara, 2024). This issue becomes particularly evident when intricate reasoning and contextual understanding are necessary to verify claims accurately (Chai et al., 2023). Several studies have explored the integration of LLMs with external knowledge sources to improve their performance in fact-checking tasks (Cui et al., 2023; Ding et al., 2023).

The incorporation of knowledge graphs into fact-checking frameworks has also garnered attention. KGs, such as DBpedia (Lehmann et al., 2015), provide structured and linked data that can enhance the contextual understanding of LLMs. Knowledge graphs have been used to improve various NLP tasks by providing additional context and relationships between entities, as demonstrated by initiatives for knowledge-aware language models (Li et al., 2023; Logan Iv et al., 2019) and KG-BERT (Yao et al., 2019).

Approximate string matching, also called fuzzy string matching, is a technique used to identify partial matches between text strings (Navarro, 2001). Fuzzy matching techniques (Navarro, 2001) have been applied to enhance the integration of LLMs and KGs (Wang et al., 2024).

The Levenshtein distance-based similarity measure (Levenshtein et al., 1966) is one such technique that helps identify strings with approximate matches, which can be useful for finding relevant connections between entities by accommodating minor discrepancies in data representation. This approach has been beneficial in refining the outputs of LLMs, ensuring that only valid and contextually appropriate connections are considered (Guo et al., 2023).

Our proposed method, FactGenius, builds on these advancements by combining zero-shot prompting of LLMs with a fuzzy relation-mining technique to improve reasoning over KGs. This methodology leverages DBpedia as a structured source of linked data to enhance fact-checking accuracy. By using LLMs to filter potential connections between entities and refining these connections through fuzzy matching, FactGenius aims to address the limitations of existing fact-checking models.

3 Preliminaries

A Knowledge Graph (KG) G is a set of triples (s, r, o), with $s, o \in E$ and $r \in R$, where E is the set of entities, and R is the set of relations connecting those entities. A KG can be viewed either as a set of triples or as a graph with nodes in E and edge labels in R. Hence, when we discuss the 1-hop neighborhood of a certain entity e, we refer to the set of entities connected to e through an edge in this graph. For a triple (s, r, o), we consider s to be connected to o through an edge labeled as r, while we consider o to be connected to s through an edge labeled as $\sim r$, where $\sim r$ denotes the inverse relation of r.

We consider natural language sentences in their intuitive sense.

Given a claim in natural language C, a KG G with entities E, and a set of entities relevant to the claim E_C, the *fact verification with KG evidence task* is to predict whether the claim C is supported or not according to the evidence in G.

4 Methodology

We introduce the FactGenius system for the fact verification with KG evidence task. Our system has two main components: a graph filtering component that selects the relevant KG evidence for the input claim, and a classifier component that uses this evidence together with the claim to predict whether the claim is supported or not.

FactGenius leverages the capabilities of a Large Language Model (LLM) to filter the set of triples in the input graph G. More concretely, an LLM is used in a zero-shot setting to select the relevant relations from the 1-hop neighborhood of the entities E_C associated with claim C. Since the output of LLMs can be erroneous, the triples are further validated against the unfiltered set using fuzzy matching techniques. Finally, the classifier, which can be fine-tuned over pre-trained models like BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019), or a Zero-Shot LLM, determines whether the claim is supported or refuted. The overall pipeline is shown in Figure 1.

4.1 FactGenius: Relation Filtering with LLM and Fuzzy Matching

The first step in our FactGenius pipeline is identifying the graph evidence relevant to the input claim. We select the relevant relations in the 1-hop neighborhood of the claim entities by employing LLM-based filtering. Once we have the relevant relations, we select the 1-hop neighborhood triples. These triples are then turned into strings and used together with the claim by the classifier.

4.1.1 LLM Prompt-Based Filtering

We utilize an LLM, specifically the Llama3-Instruct model, to identify and filter potential connections between entities based on a given claim.

This is done in the following way. First, we must select a set of relations to filter using the LLM. Given that KGs can be very large, for example, DBpedia contains billions of triples and thousands of edges (Lehmann et al., 2015), considering the full set of relations in an LLM prompt is infeasible. In FactGenius, we choose to look only at the 1-hop neighborhood of the given set of claim entities E_C to generate the initial set of relations. We therefore construct a set of 1-hop relations for each entity e, i.e. $\{r | (e, r, e_1) \in G\}$, which we will denote by $R_C(e)$. The LLM is then given the claim C and the set of relations $R_C(e)$ for each entity relevant to the input claim (each $e \in E_C$), and it outputs subsets of each $R_C(e)$, which we denote by $R_C^{llm}(e)$. A prompt example is given in Figure 2.

A retry mechanism is employed to handle potential failures in LLM responses. If the LLM output is inadequate (e.g., empty or nonsensical), the request is retried up to a specified maximum

System prompt:
You are an intelligent graph connection finder. You are given a single claim and connection options for the entities present in the claim. Your task is to filter the Connections options that could be relevant to connect given entities to fact-check Claim1. ~ (tilde) in the beginning means the reverse connection.

User prompt:
Claim1:
<<<Well, The celestial body known as 1097 Vicia has a mass of 4.1kg.»>
## TASK:
- For each of the given entities given in the DICT structure below:
Filter the connections strictly from the given options that would be relevant to connect given entities to fact-check Claim1.
- Think clever, there could be multi-step hidden connections, if not direct, that could connect the entities somehow.
- Prioritize connections among entities and arrange them based on their relevance. Be extra careful with signs.
- No code output. No explanation. Output only valid python DICT of structure:
<<<
{
`"1097_Vicia"`: ["...", "...", ...]
# options (strictly choose from): `discovered`, `formerName`, `epoch`, `periapsis`, `apoapsis`, ..., `Planet/temperature` `"4.1"`: ["...", "...", ...],
# options (strictly choose from): `~length`, `~ethnicGroups`, `~percentageOfAreaWater`, `~populationDensity`, `~engine`, ..., `~number`
}
>>>

Figure 2: Filtering prompt example. The text inside < < < and > > > changes with each input.

number of attempts, in practice 10. In our experiments, however, we did not encounter any cases where retries exceeded this limit. If the limit is exceeded, the non-filtered sets of relations are returned.

4.1.2 LLM Output Validation

As mentioned, the LLM could output relations that are not in G. That is, $R_C^{llm}(e)$ is not necessarily a subset of $R_C(e)$ or even R.

We therefore pass the LLM output through a validation stage, which has two sub-stages, namely *Stage A* and *Stage B*.

In *Stage A*, we perform validation of the relation set for each entity from the claim. That is, for each entity $e \in E_C$, we select the subset of $R_C(e)$ that best matches the LLM output $R_C^{llm}(e)$. To do so, we fuzzily match the relations in $R_C(e)$ to the relations in $R_C^{llm}(e)$ using Levenshtein distance. A threshold on this distance is considered to decide whether two relations match or not.

The limitation of the first validation type is that if the LLM suggests the correct relation, but associates it with the wrong entity, this relevant relation is removed through the first validation type. We will exemplify this on the prompt in Figure 2. The model is given the entities `1097_Vicia` and `4.1`, each with the list of possible relations. If the model identifies `Planet/temperature` but associates it with `4.1` instead of `1097_Vicia` this relation is removed during *Stage A* validation.

To address this limitation, we introduce *Stage B* validation. In this stage, we consider the full set of relations generated by the LLM for all entities associated with the input claim, i.e., $R_C^{llm} = R_C^{llm}(e_1) \cup ... \cup R_C^{llm}(e_n)$ for all $e_1, \ldots, e_n \in E_C$. Similarly to *Stage A*, we use Levenshtein distance to compare the relations in $R_C(e)$ with the filtered relations, but we consider the full filtered set R_C^{llm} instead of the entity-specific set $R_C^{llm}(e)$. The details are explained in Algorithm 1.

4.2 Claim-Driven Relation Filtering

To measure the effectiveness of LLM in relation filtering (as described in 4.1), we create a baseline that ensures only the relations most pertinent to the claim, based on lexical similarity, are selected. To filter relations relevant to a claim, we begin by tokenizing the claim sentence, excluding stop words, to obtain a list of significant word tokens. Next, for each entity $e \in E_C$ present in the claim, we gather all 1-hop relations $R_C(e)$.

Algorithm 1 LLM output validation

```
1: Input: $E_C = \{e_1, ...., e_n\}$ - entities in the claim;
2: $R_C(e_1), ..., R_C(e\_n)$: relations in the 1-hop neighborhood for each entity in the claim;
3: $R_C^{llm}(e_1), ..., R_C^{llm}(e_n)$: relation sets output by the LLM;
4: $stage$: validation stage, either A or B
5: Output: $R'_C(e_1), ..., R'_C(e\_n)$- Validated relation sets.
6: procedure VALIDATERELATION
7:     Initialize: probable_connections: { }
8:     for each $e \in E_C$ do
9:         for each $r \in R_C(e)$ do
10:            if $stage = \mathbf{A}$ then
11:                $R^{llm-compare} = R_C^{llm}(e)$
12:            else
13:                $R^{llm-compare} = R_C^{llm}(e_1) \cup ... \cup R_C^{llm}(e_n)$
14:            end if
15:            for each $r^{llm} \in R^{llm-compare}$ do
16:                $d = $ LEVENSHTEINDISTANCE$(r, r^{llm})$
17:                if $d > 90$ then
18:                    $R'_C(e) = R'_C(e) \cup \{r\}$
19:                end if
20:            end for
21:        end for
22:    end for
23: end procedure
```

We then apply a fuzzy matching process to each tokenized word in the claim, comparing it to the relations in $R_C(e)$ using the Levenshtein distance. This process yields a subset of relations $R'_C(e)$, where each relation's similarity to the claim words exceeds a predefined threshold.

4.3 With Evidence Classifier

In this configuration, the model is supplied with both the claim and graphical evidence as input, and it then makes predictions regarding the label. FactGenius utilizes graph filtering, as explained in Section 4.1, to ensure retention of the most relevant and accurate connections.

4.4 Evidence Stringification

To effectively pass evidence triples to the language model, we must first convert these triples into a string format. For each entity e in the claim with its associated relations $\{r \mid (e, r, e_1) \in G\}$ extracted from the graph G, we transform each triplet (e, r, e_1) into the string format "$\{e\} > -\{r\} - > \{e_1\}$". For multiple triples of evidence, the resulting strings are simply concatenated into a single evidence string, preserving the order and structure of the triples. This approach ensures a seamless and coherent integration of structured graph data into the language model's input.

4.5 Zero-Shot LLM as Fact Classifier

This involves utilizing Llama-3-Instruct as a fact classifier, to predict whether the given input claim and evidence string are supported or refuted. A retry mechanism is implemented to handle potential failures in LLM responses. A prompt example with evidence is shown in Figure 3.

4.6 Fine-Tuning Pre-Trained Models

Pre-trained BERT-base-uncased[1] and RoBERTa-base are fine-tuned with the claim and evidence string as inputs to predict whether the claim is supported or refuted.

An ablation study was conducted to evaluate the contributions of each stage of our approach. This involved sequentially removing Stage-B and measuring the system's performance. The results of the ablation study allowed us to quantify the impact of both stages on the overall performance of the model. Accuracy was used as an evaluation metric across all reasoning types to quantify performance improvements from the ablation study.

4.7 Implementation

Our FactGenius system implementation leverages several advanced tools and frameworks to ensure efficient and scalable processing. The Llama3-Instruct inference server is set up using vLLM (vLLM Project, 2024; Kwon et al., 2023), running on an NVIDIA A100 GPU (80 GB vRAM) to facilitate rapid inference. This server runs standalone, integrating seamlessly with the FactGenius pipeline.

For model fine-tuning and evaluation, we employ the Hugging Face Transformers library, utilizing the `Trainer` class for managing the training process. This setup allows for the fine-tuning of pre-trained models like BERT and RoBERTa within our pipeline. Hyper-parameters such as batch size, learning rate, and training epochs are configured to optimize performance, with computations accelerated by PyTorch.

The models were fine-tuned on a single NVIDIA V100 GPU, with RoBERTa requiring around 25 minutes per epoch with a batch size of 32, and BERT taking around 8 minutes per epoch with a batch size of 64. The fine-tuning process utilized the Adam optimizer with settings of $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e-6$ for RoBERTa, and $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 1e-8$ for BERT.

[1]huggingface.co/google-bert/bert-base-uncased

A weight decay of 0.01 was used over all the layers. A learning rate of $5e-6$ was used with early stopping over validation loss for 3 epochs, retaining the best epoch's weights.

5 Experiments

To evaluate the performance of our proposed methods, we conducted a series of experiments comparing different strategies for fact-checking on the FactKG (Kim et al., 2023b) benchmark.

5.1 Dataset

The FactKG dataset (Kim et al., 2023b) comprises 108,000 claims constructed using various reasoning patterns applied to facts sourced from DBpedia (Lehmann et al., 2015). Each data point consists of a natural language claim in English, the set of DBpedia entities mentioned in the claim, and a binary label indicating the claim's veracity (Supported or Refuted). The distribution across labels and five different reasoning types is shown in Table 1. The relevant relation paths starting from each entity in the claim are provided, which aids in the evaluation and development of models for claim verification tasks.

The dataset is accompanied by two processed versions of the FactKG Knowledge Graph, derived from DBpedia 2015. The first version encompasses the entire DBpedia dataset with the directionality of edges removed by incorporating reverse relation triples, denoted as *DBpedia-Full*. The second version is a curated subset of the first, containing only the relations pertinent to FactKG, thus enabling more focused and efficient analysis, and is referred to as *DBpedia-Light*.

Set	**Train**	**Valid**	**Test**
Total Rows	86,367	13,266	9,041
True (Supported)	42,723	6,426	4,398
False (Refuted)	43,644	6,840	4,643
One-hop	15,069	2,547	1,914
Conjunction	29,711	4,317	3,069
Existence	7,372	930	870
Multi-hop	21,833	3,555	1,874
Negation	12,382	1,917	1,314

Table 1: Data distribution across labels and five reasoning types.

5.2 Results

Following prior work (Kim et al., 2023b,a), we conducted experiments with two types of approaches: one that takes as input only the claim, referred to as *Claim Only*, and another that integrates KG information, referred to as *With Evidence*. The goal of this comparison is to assess whether the required knowledge is already stored in the weights of pre-trained large language models or if injecting KG information is beneficial. The results are summarized in Table 2.

5.3 Claim Only

For the *Claim Only* scenario, we compared four methods: two from the previous literature and two designed by us. We selected two of the best-performing methods from prior work: the BERT-based claim-only model introduced with the FactKG dataset by Kim et al. (Kim et al., 2023b), and the ChatGPT-based model subsequently introduced by Kim et al. (Kim et al., 2023a). Additionally, we experimented with two models of our own design: we used the Meta-Llama-3-8B-Instruct[2] (Meta, 2024) model with zero-shot prompting, and a RoBERTa-base (Liu et al., 2019) model, which we fine-tuned on the fact verification task. An example of the prompt we used for Meta-Llama-3-8B-Instruct is found in Appendix B.

Our results show that RoBERTa outperformed the reported accuracy of BERT (Kim et al., 2023b), achieving an accuracy of 0.68, which is on par with the 12-shot ChatGPT model reported in the KG-GPT paper (Kim et al., 2023a). This suggests that RoBERTa inherently stores knowledge relevant for fact-checking, at least on the FactKG benchmark. Our prompting approach, however, achieved a score of 0.61, underperforming on the task.

5.4 With Evidence

In the *With Evidence* setting, we compared different versions of our FactGenius system with two systems from prior work (Kim et al., 2023b,a). For our FactGenius approach, we experimented with five versions, using either an LLM classifier with prompting (Llama3-Instruct-zero-shot in Table 2) or a fine-tuned LLM as the classifier, either BERT-based (Devlin et al., 2019) or RoBERTa-based (Liu et al., 2019). For both the BERT-based and RoBERTa-based systems, we experimented with both *stage A* and *stage B* output validation.

[2]huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

Input type	Source	Model	One-hop	Conjunction	Existence	Multi-hop	Negation	Total
Claim Only	FactKG (Kim et al., 2023b)	BERT*	0.69	0.63	0.61	0.70	0.63	0.65
	KG-GPT (Kim et al., 2023a)	ChatGPT (12-shot)*	-	-	-	-	-	0.68
	Ours	Llama3-Instruct-zero-shot	0.61	0.67	0.59	0.61	0.53	0.61
	Ours	RoBERTa	0.71	0.72	0.52	0.74	0.54	0.68
With Evidence	FactKG	GEAR*	0.83	0.77	0.81	0.68	0.79	0.77
	KG-GPT	KG-GPT (12-shot)*	-	-	-	-	-	0.72
	Ours on DBpedia-Light	Claim-driven relation filtering	0.81	0.71	0.98	0.71	0.76	0.78
	FactGenius (Ours) on DBpedia-Light	Llama3-Instruct-zero-shot	0.72	0.75	0.76	0.62	0.52	0.68
		BERT-stage-A	0.85	0.80	0.91	0.79	0.78	0.81
		BERT-stage-B	0.85	0.83	0.88	0.81	0.73	0.82
		RoBERTa-stage-A	0.84	0.86	0.88	0.82	0.77	0.84
		RoBERTa-stage-B	**0.89**	**0.89**	**0.93**	**0.83**	**0.78**	**0.87**
	FactGenius (Ours) on DBpedia-Full	Llama3-Instruct-zero-shot	0.72	0.76	0.72	0.61	0.51	0.68
		BERT-stage-A	0.81	0.83	0.67	0.80	0.56	0.76
		BERT-stage-B	0.81	0.81	0.67	0.80	0.56	0.76
		RoBERTa-stage-A	0.86	0.85	0.91	0.79	0.82	0.84
		RoBERTa-stage-B	0.86	0.86	0.90	0.82	0.79	0.84

Table 2: Comparing our method with other strategies and methods in terms of reported accuracies in the test set. The * symbol indicates results taken directly from prior works, whereas '-' indicates results were not reported by prior works.

5.4.1 On DBpedia-Light Knowledge Graph

Our results show that adding evidence to the Llama3-Instruct model's instructions significantly improved its accuracy from 0.61 to 0.68. This indicates that even for large language models, incorporating relevant evidence can enhance fact-checking performance in a zero-shot learning scenario. However, directly applying zero-shot prompting with Llama3-Instruct did not yield superior performance compared to claim-driven relation filtering. The performance improved when using fine-tuned BERT or RoBERTa as classifiers. We also observed that the performance of the pipeline increased further when stage-B was used instead of stage-A relation mining, with fine-tuned RoBERTa performing better than BERT.

To assess the contribution of the validation stages, we applied both stages to our best-performing model, the RoBERTa-based system. We found that employing *stage A* of filtering resulted in an accuracy of 0.84. Incorporating *stage B* further improved the performance to 0.87. The second stage enhanced performance across most reasoning types, with notable improvements in conjunction and negation tasks. We achieved the highest performance by fine-tuning RoBERTa with stage-B relation mining, leading to an accuracy of 0.87 on the DBpedia-Light knowledge graph. To the best of our understanding, FactKG uses the DBpedia-Light graph, while KG-LLM employs DBpedia-Full, as inferred from their respective public implementations.

5.4.2 On DBpedia-Full Knowledge Graph

When using the DBpedia-Full knowledge graph, we observed a decrease in performance for all model variants compared to the *DBpedia-Light* setting. The Llama3-Instruct-zero-shot approach showed a similar performance gain. Fine-tuned BERT with both stage-A and stage-B maintained moderate scores, indicating stability but not improvement. RoBERTa-stage-A and RoBERTa-stage-B models achieved comparable performance at 0.84, with both stages performing similarly, indicating that stage-B processing does not significantly outperform stage-A in the more complex graphs. These results highlight the challenges associated with scaling to larger and more complex knowledge graphs.

6 Discussion

The enhanced performance of FactGenius, particularly in Conjunction, Existence, and Negation reasoning, can be attributed to its innovative combination of zero-shot prompting using large language models (LLMs) and fuzzy text matching on knowledge graphs.

The evidence-based filtering approaches revealed significant findings. The *stage-B* validation approach improves accuracy compared to *stage-A*, although the model shows only moderate performance improvement in Multi-hop reasoning. This suggests that more advanced techniques may be necessary to handle the complexity of Multi-hop reasoning effectively.

The two-step approach of filtering and validating connections proved to be especially effective. In the first step, the LLM narrows down potential connections based on the context provided by the claim, significantly reducing the search space. The second step refines these connections through fuzzy matching, ensuring that only the most relevant and accurate ones are retained. Our comparative study confirmed the importance of both steps, with the second step being particularly beneficial for Conjunction and Negation reasoning tasks.

While fine-tuned LLM models, such as BERT and RoBERTa, generally outperformed the zero-shot Llama3-Instruct model and claim-driven relation filtering, the increased graph complexity in DBpedia-Full compared to DBpedia-Light limited the gains from fine-tuning. This limitation can be attributed to the input token restrictions of BERT and RoBERTa, which truncate inputs after 512 tokens. Truncation is more likely with the larger DBpedia-Full graph, potentially excluding relevant information, thereby reducing the effectiveness of evidence-based filtering. Additionally, the similar performance between stage-A and stage-B relation mining in the full graph setting suggests that the added complexity of stage-B does not yield better accuracy, likely due to these input constraints. These observations underscore the need for architectural adaptations or preprocessing methods to more effectively handle larger datasets.

As LLM inference is a crucial component of this framework, we employed vLLM (vLLM Project, 2024) to enable rapid inference using a single NVIDIA A100 GPU. In our experiments, the LLM inference speed was approximately 15 queries per second, including retries in case of failure. This rate is feasible, especially as LLM inference continues to be optimized with the latest technologies. Embedding LLM in this framework has proven to be a sound decision.

7 Conclusion

In this paper, we introduced FactGenius, a novel method that combines zero-shot prompting of large language models with fuzzy relation mining to improve reasoning on knowledge graphs. This approach addresses several key challenges in traditional fact-checking methods. First, the integration of LLMs allows for the leveraging of extensive pre-trained knowledge in a zero-shot setting. Second, the use of fuzzy text matching with Levenshtein distance ensures that minor discrepancies in entity names or relationships do not hinder the relationship selection process, thus improving robustness.

Our experiments on the FactKG dataset demonstrated that FactGenius significantly outperforms traditional fact-checking methods and existing baselines, particularly when fine-tuning RoBERTa as a classifier. The two-stage approach of filtering and validating connections was crucial for achieving high accuracy across various reasoning types.

The findings from this study suggest that utilizing LLMs for KG evidence retrieval holds great promise for advancing fact-checking capabilities. Future work could explore applying this approach to other domains and datasets, as well as incorporating additional structured data sources to further enhance performance.

Limitations

The primary limitation of this work is that we only consider the 1-hop neighborhood when constructing the graph evidence. While this approach performs well on the FactKG benchmark, it may not capture the multi-hop reasoning required for more complex claims in other datasets or real-world scenarios. Additionally, our evaluation is limited to FactKG, restricting the generalizability of our findings. Another limitation stems from the input context limitations of the fine-tuned models and the LLMs, particularly when dealing with entities that have extensive graph connections, leading to input length constraints and necessitating truncation. Finally, we focused on zero-shot prompting with a single LLM and did not explore few-shot learning or alternative models, which might enhance performance.

Acknowledgement

This work has benefited from the Experimental Infrastructure for Exploration of Exascale Computing (eX3) at Simula, which is financially supported by the Research Council of Norway.

References

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, et al. 2023. GraphLLM: Boosting Graph Reasoning Ability of Large Language Model. *arXiv*.

Eun Cheol Choi and Emilio Ferrara. 2024. FACT-GPT: Fact-Checking Augmentation via Claim Matching with LLMs. In *WWW '24: Companion Proceedings of the ACM on Web Conference 2024*, pages 883–886. Association for Computing Machinery, New York, NY, USA.

Kelley Cotter, Julia R. DeCook, and Shaheen Kanthawala. 2022. Fact-Checking the Crisis: COVID-19, Infodemics, and the Platformization of Truth. *Social Media + Society*, 8(1):20563051211069048.

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. ChatLaw: Open-Source Legal Large Language Model with Integrated External Knowledge Bases. *arXiv*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ACL Anthology*, pages 4171–4186.

Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, Hao Yang, Andy Kaminski, et al. 2023. Integrating action knowledge and LLMs for task planning and situation handling in open worlds. *Auton. Robot.*, 47(8):981–997.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A Survey on Automated Fact-Checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Supryadi, et al. 2023. Evaluating Large Language Models: A Comprehensive Survey. *arXiv*.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, et al. 2021. Knowledge Graphs. *ACM Comput. Surv.*, 54(4):1–37.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023a. KG-GPT: A General Framework for Reasoning on Knowledge Graphs Using Large Language Models. *ACL Anthology*, pages 9410–9421.

Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. 2023b. FactKG: Fact Verification via Reasoning on Knowledge Graphs. *ACL Anthology*, pages 16190–16206.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *SOSP '23: Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626. Association for Computing Machinery, New York, NY, USA.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, et al. 2015. DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2):167–195.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Xinze Li, Yixin Cao2, Liangming Pan, Yubo Ma, and Aixin Sun. 2023. Towards Verifiable Generation: A Benchmark for Knowledge-aware Language Model Attribution. *arXiv*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, et al. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv*.

Robert L. Logan Iv, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's Wife Hillary: Using Knowledge-Graphs for Fact-Aware Language Modeling. *arXiv*.

Meta. 2024. Meta Llama 3. [Online; https://llama.meta.com/llama3].

Preslav Nakov, David Corney, Maram Hasanain, Firoj Alam, Tamer Elsayed, Alberto Barrón-Cedeño, et al. 2021. Automated Fact-Checking for Assisting Human Fact-Checkers. In *Proceedings of the Thirtieth International Joint Conference onArtificial Intelligence, {IJCAI-21}*, pages 4551–4558. International Joint Conferences on Artificial Intelligence Organization.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88.

Estela Saquete, David Tomás, Paloma Moreda, Patricio Martínez-Barco, and Manuel Palomar. 2020. Fighting post-truth using natural language processing: A review and open challenges. *Expert Syst. Appl.*, 141:112943.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. *ACL Anthology*, pages 809–819.

Juraj Vladika and Florian Matthes. 2023. Scientific Fact-Checking: A Survey of Resources and Approaches. *arXiv*.

vLLM Project. 2024. vLLM. [Online; https://github.com/vllm-project/vllm].

Yu Wang, Nedim Lipka, Ryan A. Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024. Knowledge Graph Prompting for Multi-Document Question Answering. *AAAI*, 38(17):19206–19214.

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, et al. 2024. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. *ACM Trans. Knowl. Discovery Data*, 18(6):1–32.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for Knowledge Graph Completion. *arXiv*.

Xia Zeng, Amani S. Abumansour, and Arkaitz Zubiaga. 2021. Automated fact-checking: A survey. *Lang. Linguist. Compass*, 15(10):e12438.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, et al. 2019. GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification. *ACL Anthology*, pages 892–901.

A Zero-shot fact-checking with evidence

We experimented with a language model in a zero-shot setting for fact verification including the evidence. We prompted the model with the claim and the evidence given as a list of triples — an example of the prompt is shown in Figure 3.

```
    [{
    "role": "system", "content":
    "You are an intelligent fact-checker. You are given
a single claim and supporting evidence for the entities
present in the claim, extracted from a knowledge graph.
    Your task is to decide whether all the facts in the
given claim are supported by the given evidence.
   Choose one of {True, False}, and output a one-sentence
explanation for the choice."
    },{
    "role": "user", "content":
    '''
    ## TASK:
    Now let's verify the Claim based on the evidence.
    Claim:
    <<< The celestial body known as 1097 Vicia has a
    mass of 4.1kg. >>>

    Evidence:
    <<< 1999_Hirayama -> mass -> "4.1"
    1097_Vicia -> mass -> "9.8" >>>

    # Answer Template:
    "True/False (single word answer),
    One-sentence explanation."
    '''
    }]
```

Figure 3: Example prompt given to Llama3-Instruct with evidence for zero-shot fact-checking.

B Claim-only models

A baseline is established using the Meta-Llama-3-8B-Instruct[3] (Meta, 2024) model with zero-shot prompting for claim verification, asking it to verify the claim without evidence. Through instruction prompt engineering, the model is ensured to respond with either 'true' or 'false'. A retry mechanism is implemented to handle potential failures in LLM responses. A prompt example is shown in Figure 4. The retry mechanism simply retries calling the LLM up to a fixed number of times and diverts to a default handling function if the LLM is unable to provide a proper output.

```
[{
"role": "system", "content":
"You are an intelligent fact-checker trained on
Wikipedia. You are given a single claim, and your task
is to decide whether all the facts in the given claim
are supported by your knowledge.
Choose one of {True, False}, and output a one-sentence
explanation for the choice."
},{
"role": "user", "content":
'''
## TASK:
Now let's verify the Claim based on your knowledge.
Claim:
<<< The celestial body known as 1097 Vicia has a
mass of 4.1kg. >>>

# Answer Template:
"True/False (single word answer),
One-sentence explanation."
'''
}]
```

Figure 4: Example prompt given to Llama3-Instruct without evidence for zero-shot fact-checking.
«< ... »> signs are added just to indicate that the content inside changes for each prompt.

[3]huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct