Fact or Fiction? Exploring Diverse Approaches to Fact Verification with Language Models

Anonymous ACL submission

Abstract

Recent advancements in natural language processing (NLP) have greatly improved the performance of language reasoning and generating. However, a well known shortcoming of language models is that they tend to generate information that is untrue, referred to as *hallucinations*. In order to help advance the correctness of language models, we improve the performance and the computational efficiency of models trained on classifying claims as true or false. We use the FACTKG dataset, which is constructed from the *DBpedia* knowledge graph extracted from Wikipedia. We create fine-tuned text models and hybrid models using graphs and text that significantly outperform the benchmark FACTKG models and all other known approaches, both with respect to test-set accuracy and training time. The increase in performance and efficiency stems from simplifying the methods for retrieving subgraphs, using simple logical retrievals rather than fine-tuned language models. Finally, we construct prompts to ChatGPT 4o that achieves decent performance, but without the need of fine-tuning.

1 Introduction

The field of NLP has greatly improved with the transformer architecture (Vaswani et al., 2017) and vastly scaling up model parameters and training data (Achiam et al., 2023; Bubeck et al., 2023). Large language models (LLMs) trained on a substantial part of all internet data have passed benchmarks as passing the BAR exam (Katz et al., 2024), follow precise and complex coding instructions (Bubeck et al., 2023) and perform data analysis tasks with the same performance as human experts (Cheng et al., 2023). Despite this improvement, state of the art language models still struggle with basic planning (Bubeck et al., 2023) and frequently generates false information, known as *hallucination* (Xu et al., 2024; Huang et al., 2023; Zhang et al., 2023). In order to mitigate hallucination, we believe it is crucial to be able to classify which information is correct and which is not. Therefore, we dedicate this article to explore models used for fact verification.

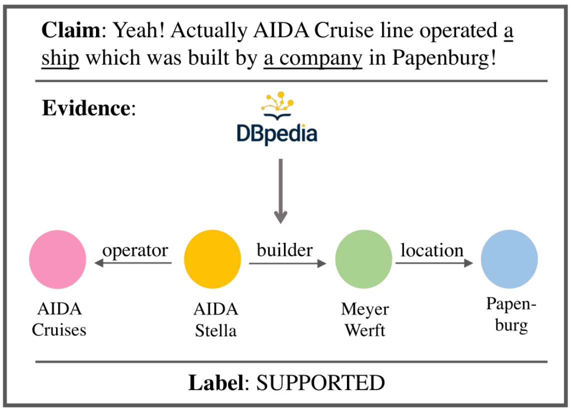

Figure 1: An example claim from FACTKG (Kim et al., 2023). The claim can be verified or refuted based on the DBpedia KG (Lehmann et al., 2015). This is Figure 1 from Kim et al. (2023).

One way of structurally working with knowledge is with *knowledge graphs* (KGs). They consist of nodes and edges linked together to represent structural concepts. The *DBpedia* KG (Lehmann et al., 2015) is a large KG extracted from Wikipedia. Nodes represent entities, such as persons, things or events, and edges represent relations, conveying how entities are related, as shown in Figure 1. For instance, a node can be the company *Meyer Werft*, and since it is located in the city *Papenburg*, they are connected with the edge *location*. We refer to *Meyer Werft, location, Papenburg* as a *knowledge triple*.

We propose models trained on FACTKG (Kim et al., 2023), a dataset proposed to better utilise knowledge graphs with fact verification (see Figure 1). It consists of 108 000 English claims that are extracted from the DBpedia knowledge graph.

About a third of the claims are manually written, while the rest are generated from the written claims to be in a colloquial form by a language model. The train and validation datasets are equipped with relevant subgraphs for each claim, which one can use to train subgraph retrieval.

In order to work with fact verification, we will work with three main model architectures:

- **Textual Fine-tuning:** Fine-tuning pretrained encoder models on text data for claim verification. We use BERT (Devlin et al., 2018) by concatenating the claims with subgraphs represented as strings.

- **Hybrid Graph-Language Model:** Using a modification of a *question answer graph neural network* (QA-GNN) (Yasunaga et al., 2021), which both uses a pretrained encoder model to embed the claim, and a GNN to structurally process the subgraphs.

- **LLM Prompting:** Deploying state-of-the-art language models in a few-shot setting, without the need for additional finetuning. We use ChatGPT 4o (Achiam et al., 2023; Open AI, 2024) for this setting.

We selected these three approaches to explore a variety of different models used in NLP, and compare how they perform on fact verification. The text-based finetuning, which is a widely used technique, serves as a conventional method. The QA-GNN architecture is a more specific model for this task, that can efficiently process graph data. We explore various ways to retrieve relevant subgraphs that do not require training of language models, to make the QA-GNN train even more efficient. In contrast, the LLM prompting displays how general purpose language models can be used for fact verification, without the need of further training.

By utilising efficient subgraph retrieval methods, we are able to substantially increase the test-set accuracy on FactKG from 77.65% (Kim et al., 2023) to 93.49%. To the best of the authors knowledge, this is the best performance achieved so far on the dataset. Additionally, our models train quicker, taking only 1.5-10 hours, compared to the 2-3 days spent on the benchmark model from Kim et al. (2023), reported by the authors.

2 Related Work

2.1 The FactKG Dataset

The FactKG dataset (Kim et al., 2023) consists of 108 000 English claims for fact verification, where the downstream task is to predict whether the claim is true or false. The claims are constructed from the DBpedia KG (Lehmann et al., 2015), which is extracted from Wikipedia and represents how entities are related to each other.

The claims are constructed on either of the following five reasoning types:

- **One-hop:** To answer a one-hop claim, one only needs to traverse one edge in the KG. In other words, only one knowledge triple is needed to verify the validity of the claim.

- **Multi-hop:** As opposed to one-hop claims, one needs to traverse multiple steps in the KG to verify multi-hop claims.

- **Conjunction:** The claim includes a combination of multiple claims, which are often added together with the word *and*.

- **Existence:** These claims state that an entity has a relation, but does not specify which entity it relates to.

- **Negation:** The claim contains negations, such as *not*. The generation process varies depending on the reasoning type of the claim.

The dataset is split in a train-validation-test set of proportion 8:1:1. The train and validation set includes relevant subgraphs for each claim, but not the test set. All claims include a list of entities present in the claim and the KG.

2.2 Question Answer Graph Neural Networks (QA-GNNs)

The QA-GNN (Yasunaga et al., 2021) is a hybrid language and GNN model that both uses a pretrained language model to process the text, and couples it with a GNN reasoning over a subgraph. It is given text and a subgraph as input. The text, consisting of a question and possible answers, is added as a node to the subgraph. The language model embeds the text, and assigns a relevance score to each node in the subgraph. The relevance scores are multiplied with the node features, before being sent into the GNN. The GNN output, text-node and the text embedding are concatenated before being put into the classification layer.

3 Methods

3.1 Efficient Subgraph Retrieval

We experiment with different ways of retrieving a relevant subgraph for each claim, focusing on computational efficiency. Each datapoint in the FACTKG dataset consists of a claim and a list of entities that appear both in the claim and the KG. Since the part of DBpedia used in FactKG is fairly large (1.53GB), it is necessary to only use a small subgraph of it as input to the models. The benchmark model from Kim et al. (2023) uses two language models to predict the relevant edges and the depth of the graph. We wish to simplify this step in order to reduce the model complexity, and propose non-trainable methods for subgraph retrieval.

We experiment with the following methods (examples can be found in Figure 2):

- **Direct:** Only includes knowledge triples where both nodes are present in the entity list.

- **Contextualized:** First, include all direct subgraphs. Additionally, lemmatize the words in the claim and check if the nodes in the entity list have any relations corresponding to the lemmatized words in the claim. Include all knowledge triples where at least one node is in the entity list and the relation is found in the claim.

- **Single-step:** Includes every knowledge triple one can be traversed in one step from a node in the entity list. In other words, include every knowledge triple that contains at least one node in the entity list.

3.2 Finetuning BERT

We use BERT (Devlin et al., 2018) as our pretrained language model. We first train a baseline model using only the claims and no subgraphs, and then with all of the different methods for retrieving subgraphs. The subgraphs are converted to strings, where each knowledge triple is represented with square brackets, and the name of the nodes and edges are the same as they appear in DBpedia. The order of the knowledge triples is determined by the order of the list of entities in the FactKG dataset and the order of the edges in DBpedia. The subgraphs are concatenated after the claims and a " | " separation token.

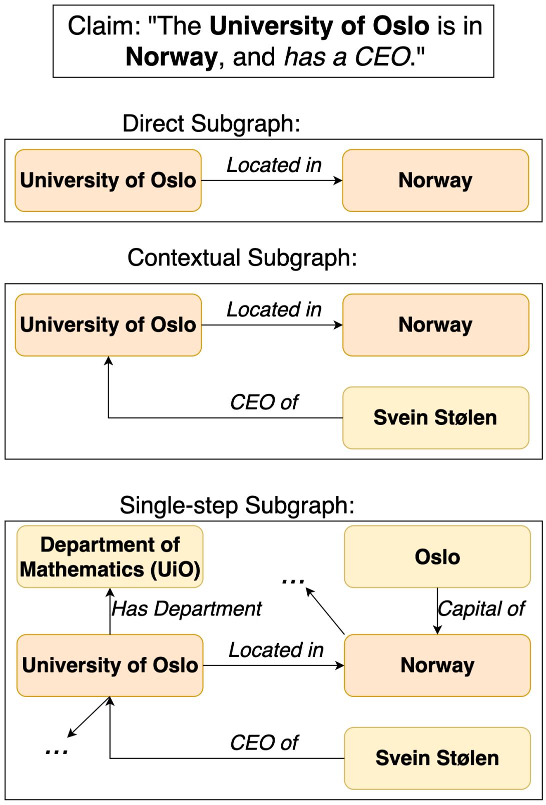

Figure 2: **Examples of the different subgraphs explored in this article**. Boxes and **bold letters** represent entities, while arrows and *italic letters* represent relations. This claim is meant for illustrative purposes and is not present in the FACTKG dataset.

3.3 QA-GNN Architecture

In order to adapt the QA-GNN to the fact verification setting, we perform some slight modifications. Because the possible answers are always "true" or "false", we embed only the claims, instead of the question and answer combination. Additionally, we do not connect the embedded question or claim to the subgraph.

We use a pre-trained BERT (Devlin et al., 2018) model as the language model to embed and calculate the relevance scores. In order to reduce the complexity of the model, we use a frozen BERT to calculate the embeddings for the nodes and the edges in the graph. This way, all of the embeddings in the graph can be pre-calculated. We use the last hidden layer representation of the CLS token, which is of length 768. The BERT that calculates the relevance scores and the embedding of the claim is not frozen. The relevance scores are computed as the cosine similarity between the claim embedding and the embedding of the text in the nodes.

We use a graph attention network (Veličković et al., 2017) for our GNN. Since the subgraphs are quite shallow, we only use two layers in the GNN, and apply batch norm (Ioffe and Szegedy, 2015). Each layer has 256 features, which is mean-pooled and concatenated with the BERT embedding and sent into the classification layer. We add dropout (Srivastava et al., 2014) to the GNN layers and the classification layer.

3.4 ChatGPT Prompting

We construct a prompt for ChatGPT 4o in order to answer a list of claims as accurately as possible. This is done by creating an initial prompt and validating the results on 100 randomly drawn claims from the validation set, and by trying different configurations of the prompt until we do not get a better validation set accuracy. We then use the best prompt with 100 randomly drawn unseen test-set questions, and attempt to ask 25, 50 and 100 claims at a time, to see if the amount of claims at a time influences the performance. All the experiments are run three times.

Since we do not have access to vast enough computational resources to run an LLM, this analysis is limited by only using 100 datapoints from the test set. In order to get access to a state-of-the-art LLM, we used the ChatGPT website with a "ChatGPT Plus" subscription to perform the prompting. This model is not seeded, so the exact answers are not reproducible, but every prompt and answer are available in the software provided with this article. We used the ChatGPT 4o model 30th of May 2024. Every prompt was performed in the "temporary chat" setting, so the model did not have access to the history of previous experiments.

Due to the inability to use the entire test set and the lack of reproducibility, we do not directly compare this experiment to the other models. However, we still believe it serves as a valuable benchmark. Recently, the performance of LLMs has rapidly improved, which suggests that their applications will continue to broaden. Additionally, this approach is not fine-tuned, and may work as an interesting benchmark that can contextualise the results of the other models.

3.5 Benchmark Models

We will compare the results against the best benchmark models from (Kim et al., 2023) and the best performing models known to the authors, found in (Gautam, 2024). These comparisons include both baselines that use only the claims and models that also incorporate subgraph evidence.

Claim-Only Models:

- **FactKG BERT Baseline**: The baseline model from (Kim et al., 2023) uses a fine-tuned BERT, training only on the claims.

- **RoBERTa Baseline**: Similarly to above, the baseline from (Gautam, 2024) uses a fine-tuned language model with claims only, but uses RoBERTa (Liu et al., 2019) as the base model.

Models Utilising Subgraphs:

- **GEAR-Based Model**: The benchmark model from (Kim et al., 2023) is inspired by GEAR (Zhou et al., 2019), but has been adapted to handle graph-based evidence. It uses two fine-tuned language models to retrieve the subgraphs. One of them predicts relevant edges, the other predicts the depth of the subgraph.

- **FactGenius:** This model (Gautam, 2024) combines zero-shot LLM prompting with fuzzy text matching on the KG. The LLM filters relevant parts of the subgraphs, which are then refined using fuzzy text matching. Finally, a fine-tuned RoBERTa is used to make the final prediction.

3.6 Further Details

Due to computational constraints, we tuned the hyperparameters one by one, instead of performing a grid search. The training was performed on a RTX 2080Ti GPU with 11GB vRAM. The BERT model has 109.483.778 parameters, which all were fine-tuned. The QA-GNN used a total of 109.746.945 parameters. The FactKG dataset comes with a lighter version of DBpedia that only contains relevant entries, which was used for this paper. Further details can be found in Appendix A.

4 Results: Improved Performance and Efficiency

The test results for our best model configurations and the benchmark models can be found in Table 1. The best performing model is the fine-tuned BERT, followed by the QA-GNN and the benchmark models. The fine-tuned BERT without any subgraphs were able to achieve higher performance than the one from (Kim et al., 2023).

Input Type	Model	One-hop	Conjunction	Existence	Multi-hop	Negation	Total
Claim Only	FactKG BERT Baseline	69.64	63.31	61.84	70.06	63.62	65.20
	FactGenius RoBERTa Baseline	71	72	52	74	54	68
	BERT (no subgraphs)	67.71	67.48	62.51	73.28	64.23	68.99
With Subgraphs	FactKG GEAR Benchmark	83.23	77.68	81.61	68.84	79.41	77.65
	FactGenius RoBERTa-two-stage	89	85	95	75	87	85
	QA-GNN (single-step)	79.08	74.43	83.37	74.72	79.60	78.08
	BERT (single-step)	**97.40**	**97.51**	**97.31**	**80.32**	**92.54**	**93.49**

Table 1: **Test-set accuracy for the best models from this article and the best benchmark models.** The FactKG models are from (Kim et al., 2023), while the FactGenius models are from (Gautam, 2024). The fine-tuned BERT model performed the best, while the QA-GNN was the computationally most efficient model.

Additionally, our models were much faster to train. While the GEAR model used 2-3 days to train on an RTX3090 GPU (reported by the authors by email), our QA-GNN only used 1.5 hours. The training time of our fine-tuned BERT model was greatly influenced by the size of the subgraphs we used. With no subgraphs, it took about 2 hours to train, while with the one-hop subgraph it took 10 hours. FactGenius was reported to use substantially more computational resources, running the LLM inference on a A100 GPU with 80GB vRAM for 8 hours.

4.1 Successful Subgraphs Retrievals

We now look at the different configurations for the subgraph retrievals, which greatly influenced the performance of the models. Since the *direct* and *contextual* approach only includes subgraphs if a certain requirement is fulfilled, it will result in some of the claims having empty subgraphs. In the training and validation set, 49.0% of the graphs were non-empty for the *direct* approach, and 62.5% were non-empty for the *contextual* approach. The *single-step* method resulted in vastly bigger subgraphs.

While the QA-GNN could handle the big subgraphs efficiently, the fine-tuned BERT was severely slowed down when the size of the subgraphs got bigger. Therefore, we substituted any empty subgraphs with the *single-step* subgraph when using QA-GNN, but kept the empty graphs when using fine-tuned BERT.

The results can be found in Table 2. We see a clear improvement in BERT when using the direct subgraphs over none, a small improvement when using the contextual subgraphs, and a big improvement when using the single-step method. The same is true for the QA-GNN, but the differences in performance are smaller.

Since we used non-trainable subgraph retrieval methods and a frozen BERT for embedding the nodes and edges in the subgraphs, we performed this processing before training the models. During training, the models used a lookup table to get the subgraphs and the word embeddings, which significantly decreased the training time. The retrieval of all the subgraphs took about 15 minutes, and the embedding of all the words appearing in them took about 1 hour. We also tried training a QA-GNN without frozen embeddings, but it ran so slow that we were not able to carry out the training with our available computational resources.

4.2 ChatGPT Works Better when Asking for Explanations

The results for the ChatGPT prompting can be found in Table 3. The accuracy is substantially lower than from our best models, but better than the baselines using only the claims. The accuracy is fairly consistent over the three runs, and we do not see a big difference between the amount of questions asked at a time.

We started with an initial prompt asking for just the truth values for a list of claims, and updated it to also include some training examples and to ask for explanations. Several configurations of the prompt were tested, and it was also improved based on feedback from ChatGPT.

We saw the biggest improvement when we asked for a short explanation of the answers, instead of just the truth values. Without asking for explanations, the amount of answers were often longer or shorter than the amount of questions, but this never happened when explanations were included. We added numbers to the questions to further help with this issue. We also saw a slight improvement by formulating the prompt with bullet point lists and by including some example inputs and outputs from the training set. The final prompt can be found in Figure 3.

Model	One-hop	Conjunction	Existence	Multi-hop	Negation	Total
BERT (no subgraphs)	67.71	67.48	62.51	73.28	64.23	68.99
BERT (direct)	80.24	83.30	59.05	77.62	74.58	79.64
BERT (contextual)	81.20	84.45	61.05	77.04	77.40	80.25
BERT (single-step)	**97.40**	**97.51**	**97.31**	**80.32**	**92.54**	**93.49**
QA-GNN (direct)	74.60	74.01	58.97	76.41	74.12	75.01
QA-GNN (contextual)	76.58	69.94	84.68	74.58	80.75	76.12
QA-GNN (single-step)	79.08	74.43	83.37	74.72	79.60	78.08

Table 2: **Test-set accuracy for different subgraph retrieval methods on FactKG.** The *direct* approach only includes knowledge triples where both nodes appear in the claim, the *contextual* also includes edges appearing in the claim, and the *single-step* includes all knowledge triples where at least one node appear in the claim. The QA-GNN models use the single-step subgraph if the direct or contextual is empty, while the BERT does not.

Model	Accuracy (mean ± std)
ChatGPT 25 questions	73.67 ± 0.5
ChatGPT 50 questions	**76.33** $\pm$ 3.3
ChatGPT 100 questions	73.00 ± 1.4

Table 3: **Test-set accuracy for different configurations of ChatGPT prompting.** The metrics are averaged over three runs. The prompts included 25, 50 or 100 claims at a time, but the same claims were used in all of the configurations.

5 Discussion

We were able to train better and more efficient models by simplifying the subgraph retrieval methods, both by using a fine-tuned BERT and a slightly modified QA-GNN model. While the QA-GNN models trained the fastest, the fine-tuned BERT clearly performed the best, significantly outperforming the benchmark models.

All of the models performed better the bigger the subgraphs were. This suggests that the model architectures are able to utilise the relevant parts of the subgraphs, without needing an advanced subgraph retrieval step. This is emphasised by our fine-tuned BERT model achieving a 93.49% test set accuracy by only using the single-step subgraphs, while the GEAR model from (Kim et al., 2023), which trained two language models to perform graph retrieval, achieved a 77.65% test-set accuracy.

One possible limitation of our subgraph retrial methods is that they never include more than one step away from an entity node, while the trained approach from Kim et al. (2023) is dynamic and may include more. This might make the hypothesis that the simple subgraph retrieval methods will perform worse on *multi-hop* claims than the dynamically trained, however, we see the exact opposite behaviour. The best BERT and QA-GNN models score 80.32% and 74.72% at the multi-hop claims respectively, while the dynamic benchmark model scores 68.84%, even lower than the models not using the subgraphs at all. We do however see that the best performing BERT model clearly performs the worst on the multi-hop claims compared to the other claim types, indicating that even bigger subgraphs might be beneficial.

While the sample size for the ChatGPT metrics were small, it does indicate that non-fine-tuned LLMs can achieve adequate few-shot performance. The performance does not seem to be substantially compromised when the amount of questions prompted increases, so with a bigger access to computational resources, it might be possible to prompt the full test-set at once. The removal of fine-tuning greatly improves the ease of use if one only needs to verify a few claims. Therefore, despite not performing as well as the trained model, this approach could be useful if the performance of LLMs continues to improve.

6 Conclusion and Future Work

Our results show that with simple, yet efficient methods for subgraph retrieval, our models were able to improve with respect to both performance and efficiency. The fine-tuned BERT model with single-step subgraphs clearly achieves the best performance, while the QA-GNN models are more efficient to train.

This indicates that complex models can work well with simple subgraph retrieval methods. Since the single-step subgraphs mostly contain information not relevant for the claims, the model is itself able to filter away irrelevant information, and complex subgraph retrieval methods may hence not be necessary for accurate fact verification. Additionally, since the best performing model performed

Task:
Determine the truth value (True or False) of the following claims based on information verifiable from Wikipedia, as represented in the DBpedia knowledge graph. Provide your answers without using real-time internet searches or code analysis, relying solely on your pre-trained knowledge.
Instructions:

- You will evaluate the following claims, presented one per line.
- Base your answers solely on your knowledge as of your last training cut-off.
- Provide answers in Python list syntax for easy copying.
- Respond with `True` for verifiable claims, and `False` otherwise.
- Include a brief explanation for each answer, explaining your reasoning based on your pre-training.
- If the claim is vague or lacks specific information, please make an educated guess on whether it is likely to be True or False.

Output Format: Format your responses as a list in Python. Each entry should be a tuple, formatted as (claim number, answer, explanation).
Example Claims:
1. The Atatürk Monument is located in Izmir, Turkey, where the capital is Ankara.
2. Yes, Eamonn Butler's alma mater is the University of Texas System!
3. I have heard 300 North LaSalle was completed in 2009.
4. The band Clinton Gregory created an album in the rock style. ...
Example output:
[
(1, True, "The Atatürk Monument is indeed located in Izmir, and the capital of Turkey is Ankara."),
(2, False, "Eamonn Butler did not attend the University of Texas System; he is a British author and economist whose educational background does not include this institution."),
(3, True, "300 North LaSalle in Chicago was indeed completed in 2009."),
(4, False, "Clinton Gregory is primarily known as a country music artist, not rock."),
...
]
Here are the actual claims you should answer:

Figure 3: **Final prompt used to get truth values from ChatGPT 4o**. The actual questions are not included, but were in the format of the **Example Claims**. The **Example Claims** are from the training set, and the **Example Output** is copy pasted from an actual ChatGPT answer.

the poorest with *multi-hop* claims, future research could explore simple subgraphs retrieval methods allowing for bigger depths than one. Future work should also be directed towards running similar experiments on other datasets.

We also encourage researchers that have access to bigger computational resources to further explore the performance of LLMs for fact verification. A core limitation of our ChatGPT prompting was the inability to use the full test-set, and we consider this crucial for further development. We also think it would be especially interesting to make LLM and KG hybrid models. Since our results indicate that simple single-step subgraph retrievals outperform more complex methods, a promising path of future research would be to simply use both the claims and the single-step subgraphs as input to the LLM. If possible, the LLM could also be fine-tuned on the dataset. We also encourage future work to create fully reproducible results with LLMs, which we were unable to do.

7 Limitations

Our experiments with ChatGPT were done on a small sample of test questions, with a model that was not possible to seed, and therefore is not reproducible. Due to the small sample size, we are not able to directly compare the performance to other approaches. The lack of reproducibility, which stems from the state-of-the-art model that was available to the author is not fully publicly available, makes it impossible for other researchers to completely verify our findings. Finally, the process for creating prompts were not standardised, we simply tried different configurations based on our own experience with using LLMs until we could not increase the validation accuracy further. Due to these limitations, one should therefore be very hesitant to make any conclusions based on the experiments we performed with ChatGPT.

Because our intention was to compare different language models' abilities of fact verification with knowledge graphs on the FACTKG dataset, we did not conduct any experiments on other datasets. It is possible that our results will not be consistent with other datasets.

Additionally, our selection of models and hyperparameter settings could be more diverse. Due to computational constraints, we did not perform a grid search for hyperparameters, but tuned hyperparameters one by one. Which parameters we searched for were not decided in advance. A predefined grid search might lead to a fairer and more reproducible approach.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Liying Cheng, Xingxuan Li, and Lidong Bing. 2023. Is gpt-4 a good data analyst? *arXiv preprint arXiv:2305.15038*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sushant Gautam. 2024. Factgenius: Combining zero-shot prompting and fuzzy relation mining to improve fact verification with knowledge graphs. *arXiv preprint arXiv:2406.01311*.

Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. 2020. Array programming with numpy. *Nature*, 585(7825):357–362.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. 2024. Gpt-4 passes the bar exam. *Philosophical Transactions of the Royal Society A*, 382(2270):20230254.

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

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

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

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Open AI. 2024. Hello gpt 4o. https://openai.com/index/hello-gpt-4o/, Accessed 30.05.2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. Gear: Graph-based evidence aggregating and reasoning for fact verification. *arXiv preprint arXiv:1908.01843*.

Model	Learning Rate	Batch Size	Best Epoch
BERT (no subgraphs)	1e-4	32	6
BERT (direct)	1e-4	32	7
BERT (contextual)	5e-5	8	7
BERT (single-step)	5e-5	4	7
QA-GNN (direct)	1e-4	128	8
QA-GNN (contextual)	5e-5	64	17
QA-GNN (single-step)	1e-5	128	20

Table 4: **Final hyperparameters for the different models.** The direct QA-GNN model used GNN and classifier dropout rates of 0.3 and 0.3, while both the two other QA-GNN used 0.1 and 0.5.

A Hyperparameter Details

We used an AdamW optimizer (Loshchilov and Hutter, 2017) and a linear learning rate scheduler with 50 warm up steps. We used the model from the epoch with lowest accuracy loss. The hyperparameters were tuned in a line search, first testing different learning rates, and testing all the other hyperparameters with the best learning rate. We searched for learning rates in $\{1e-3, 5e-4, 1e-4, 5e-5, 1e-5\}$ for all models. We initially set the batch size to 32, except for the BERT models with large subgraphs, which were set to 4 due to memory constraints. After finding the learning rate, we tried batch sizes in $\{32, 64, 128, 256\}$. For the QA-GNN model, we initially set the GNN dropout and the classifier dropout to 0.3, and tried values in $\{0, 0.1, 0.3, 0.5, 0.6\}$. We also tried to freeze some of the layers in the base model, and to use a RoBERTa model instead of BERT, but neither of these approaches approved the validation loss.

The final hyperparameters can be found in Table A.

B Scientific Artifacts

We conducted the experiments using many python libraries, including PyTorch version 2.0.1 (Paszke et al., 2019) with CUDA version 11.7, HuggingFace Transformers (Wolf et al., 2020), NumPy (Harris et al., 2020), SpaCy (Honnibal and Montani, 2017) and NLTK (Bird et al., 2009). We will make all the code used for this paper publicly available.