FLEEK: Factual Error Detection and Correction with Evidence Retrieved from External Knowledge

Farima Fatahi Bayat[1]*, **Kun Qian**[2], **Benjamin Han**[2], **Yisi Sang**[2], **Anton Belyi**[2], **Samira Khorshidi**[2], **Fei Wu**[2], **Ihab F. Ilyas**[2], **Yunyao Li**[2]
University of Michigan - Ann Arbor[1], Apple[2]
farimaf@umich.edu, {kunqian, ben.b.han, yisi_sang, a_belyy}@apple.com
{samiraa, fwu7, iilyas, yunyaoli}@apple.com

Abstract

Detecting factual errors in textual information, whether generated by large language models (LLM) or curated by humans, is crucial for making informed decisions. LLMs' inability to attribute their claims to external knowledge and their tendency to hallucinate makes it difficult to rely on their responses. Humans, too, are prone to factual errors in their writing. Since manual detection and correction of factual errors is labor-intensive, developing an automatic approach can greatly reduce human effort. We present FLEEK, a prototype tool that automatically extracts factual claims from text, gathers evidence from external knowledge sources, evaluates the factuality of each claim, and suggests revisions for identified errors using the collected evidence. Initial empirical evaluation on fact error detection (77-85% F1) shows the potential of FLEEK. A video demo of FLEEK can be found at https://youtu.be/NapJFUlkPdQ.

1 Introduction

While textual information offers a convenient and efficient means of communication, it is critical to acknowledge its potential for misuse or unintended consequences. False or misleading information spreads easily over online platforms (Webb et al., 2016). Additionally, the emergence of powerful Large Language Models (LLMs) such as GPT models [1], Vicuna (Chiang et al., 2023), and Alpaca (Taori et al., 2023) have introduced a new avenue for knowledge-seeking inquiries. These models, however, have a tendency to hallucinate and provide creative and fluent responses that are not factually accurate (Pan et al., 2023). The limitation of LLMs to attribute their responses to external valid evidence makes it challenging to trust their responses. Therefore, having a robust fact-checking mechanism is of paramount importance to ensure the integrity and accuracy of information.

Previous works (Zhong et al., 2019; Liello et al., 2022; Liu et al., 2020) and systems like FACTGPT [2], typically formulates the fact verification as a classification task where the input consists of the evidence sentence(s) and the claim, and the output is a label indicating the veracity of the entire claim as SUPPORTED, REFUTED, or IRRELEVANT. As a concrete example, if the claim is "*United States is in North America and has 51 states*", then a sentence-level classification task would classify this claim as incorrect since there are *50* states in the United States. However, this claim actually contains one valid sub-claim: "*United States is in North America*" ✓, and one false sub-claim: "*United States has 51 States* ✗. Providing a single label stating that this claim is not supported or a single score indicating its factual accuracy is not helpful. It would still require users to manually identify text spans corresponding to potential incorrect facts, generate search queries to gather evidence from the open web, and ultimately make a decision based on multiple pieces of evidence.

In this work, we present FLEEK (**F**actua**L** **E**rror detection and correction with **E**vidence Retrieved from external **K**nowledge), an intelligent and model-agnostic tool designed to support end users (e.g. human graders) in fact verification and correction. Our tool features an intuitive and user-friendly interface, capable of automatically identifying potential verifiable facts from input text. It generated questions for each fact and queries both curated knowledge graphs and the open web to collect evidence. Our tool then verifies the correctness of the facts using the gathered evidence, and suggests revisions to the original text.

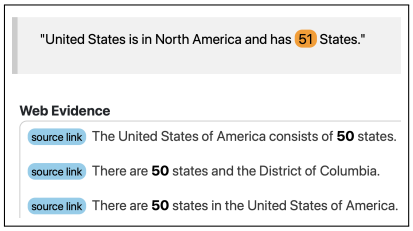
Our verification process is naturally interpretable since the extracted facts, generated questions, and retrieved evidence all directly reflect which infor-

*Work done while the author was an intern at Apple.
[1] https://platform.openai.com/docs/models
[2] https://factgpt-fe.vercel.app/

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 124–130
December 6-10, 2023

mation units contribute to the verification process. For the example mentioned above, FLEEK would highlight verifiable facts with different colors indicating their factuality levels (see Figure 1(a)), and these clickable highlights can open a dialog that further lists evidence retrieved to support or refute each claim (see Figure 1(b)).

(a) Factuality annotations by FLEEK



(b) A clickable questionable fact

Figure 1: Screenshots of FLEEK

To the best of our knowledge, FLEEK is the first verification and correction system that provides fact-level decisions, attributes them with evidence from online sources of information, and proposes factual revisions.

2 Methodology

Figure 2 shows the overall architecture of FLEEK. Basically, FLEEK can perform two tasks: *Fact verification* and *Fact Revision*. Next, we describe the methodology used to enable the two tasks.

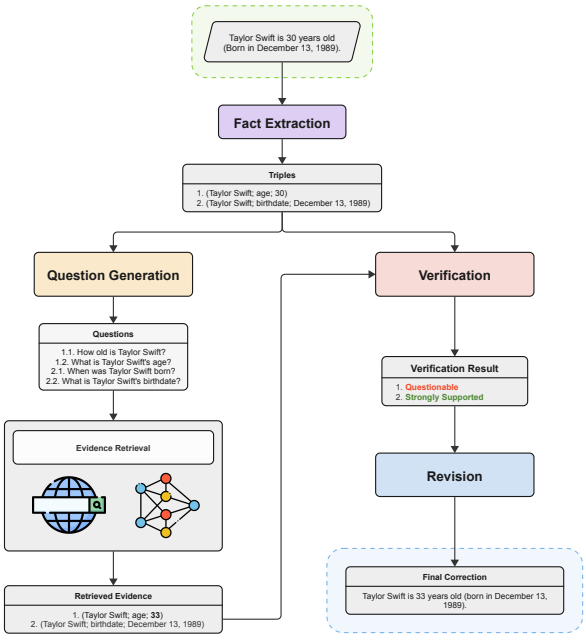


Figure 2: FLEEK verification and revision framework.

2.1 Fact Verification

Given an input passage p, we split it into a set of sentences $\{s_1, ...s_i\}$. We then verify each sentence using the sequential pipeline described below.

2.1.1 Fact Extraction

In this work, we define a *fact* as a unit of information that (1) describes a certain entity or (2) captures the relation between two entities (3) describes an event. Each fact consists of a subject, a predicate, and at least one object. We use the semi-structured triple format to represent such a fact. Our goal is to break a sentence into a set of triples such that each triple represents a verifiable piece of information. This way, we can provide more fine-grained verification details for each sentence. To exhaustively extract facts, we consider two triple formats:

Flat Triple: For binary predicates, i.e., predicates with one object, we represent the fact in the form of *(Subject; Predicate; Object)*. For example, the triple representation of the fact *"Taylor Swift is 30 years old."* is *(Taylor Swift; age; 30 years old)*.

Extended Triple Ilyas et al. (2022): For n-ary predicates where $n > 2$, i.e., predicates with multiple objects, we utilize the *extended triple* format to capture the relations between fact constituents. The extended triple format is *(Subject; Predicate; Predicate_ID; Predicate_attribute; Object)* where *Predicate_ID* is an artificial predicate identifier, *Predicate_attribute* is the name of the predicate's attribute, and *Object* is the attribute's value. For instance, the representation of the sentence *"Taylor Swift moved to Nashville at the age of 14."* is:
(Taylor Swift; moved; move_ID; place; Nashville)
(Taylor Swift; moved; move_ID; age; 14).

For an input sentence s_i, the task of Fact Extraction is to extract flat triples $T_f = \{t_{f_1}, ..., t_{f_m}\}$ and extended triples $T_e = \{t_{e_1}, ..., t_{e_n}\}$. These triples are extracted from the sentence using an open information extraction format, with each triple representing a single predicate attribute. The final output of this component is: $T = T_f \cup T_e$.

To extract these triples, we came up with five challenging human demonstrations such that, for an input sentence, they include different combinations of flat and extended triples. We prompt two instructable LLMs to obtain such triples. More details on LLMs utilized for this task, along with an in-depth analysis of the errors they generate, is provided in section 4.

2.1.2 Question Generation (QGen)

Given the output of the Fact Extraction component T, the task of Question Generation is to generate questions for each $t \in T$ such that the answer to the question is the *Object* part of t. In this way, various answers retrieved from different sources can be used to verify each triple t. Depending on the format of triple t (flat or extended), we introduce two different question generation paradigms.

Type-aware Question Generation (TQGen). Consider the triple *(Taylor Swift; birthdate; 1989)*. If the output of the QGen component is: *"When was Taylor Swift born?"*, the retrieved evidence would probably be the exact birthdate. To generate a question as specific and close to the answer as desired, we propose a type-aware question generation approach. Using TQGen, the generated question will be: *"In which year was Taylor Swift born?"*. To this end, we adopt the *Chain-of-Thought* paradigm. This involves two steps: we instruct our model to first find the *"type"* of the *Object* in the input triple, and then generate a question conditioned on the obtained type information. TQGen guides the subsequent information retrieval component to target and retrieve the exact fact that we aim to verify. Prompting LLMs with two human demonstrations was sufficient for this task.

Context-driven Question Generation (CQGen). In addition to generating a precise type-aware question, we need to provide context for extended triples so that the retrieved evidence corresponds to the exact situation that requires verification. Consider the extended triples mentioned earlier,
(Taylor Swift; moved; move_ID; place; Nashville)
(Taylor Swift; moved; move_ID; age; 14)
where the focus is on generating a question for the first triple. If we only feed the first triple to QGen, the output would not consider the time when the relocation happened. To generate a context-driven question, we need to also feed the second triple, the *context triple*, to Context-driven QGen (CQGen). The output of CQGen in this example is *"To which city did Taylor Swift move to at the age of 14?"*. For this task, we prompt LLMs with two examples.

2.1.3 Evidence Retrieval

The generated questions will be sent to two retrieval systems: a knowledge graph (KG)-based system and a web-based system.

Knowledge Graph-based: We send the question generated for each triple t to our KG question answering (KGQA) system and collect the retrieved short answers. The answer and can either be a single value (e.g., birth date, birthplace) or a list (e.g., profession, spouses). The ensuing entailment decision is derived differently for these two forms of answers (more details in Section 2.1.4).

Web-based: Similarly, we also submit the same question(s) to our web search engine (Web Search). We then take the top-k (e.g., 5) web passages returned for each question and combine them to create a consolidated set of answers. Additionally, Web Search is able to highlight the short answer a for each retrieved passage p. The final retrieval list from Web Search is in the format $[(p_1, a_1), (p_2, a_2), ..., (p_k, a_k)]$.

2.1.4 Verification

Given the triple representation of a fact t, the set of KG answers $A_{kg} = \{a_1, a_2, ...\}$, and the set of Web answers $A_w = \{(p_1, a_1), (p_2, a_2), ..\}$, the task is to decide whether t is supported by the set of retrieved evidence. This involves two steps:

Step 1 - Verify against KG answers. Based on our observation, when the evidence retrieved from the KG is a singular value, the expected answer to the question is most likely to also be a single value (e.g. city of birth). Therefore when $|A_{kg}| = 1$, we classify the fact as *"Strongly Supported"* if it is entailed by the answer, and *"questionable"* otherwise. However, if the KG answer is a list, we classify each answer in A_{kg} as either *"supporting"* or *"not supporting"* based on whether it entails the fact. In this case, due to the limited coverage of facts in KG (Dong et al., 2014; Peng et al., 2023), we verify the fact t against web answers as well.

Step 2 - Verify against Web answers. In case the KG answer is empty or a list, web answers will be also used to make a decision. We classify the answers in A_w as either *"supporting"* or *"not supporting"* evidence. Finally, the fact is labeled as *"Likely Supported"* if our system finds at least one *"supporting"* evidence and *"Questionable"* otherwise. In what follows, we describe how perform evidence classification.

Triple Entailment. For every triple t, we have a set of retrieved answers $A = A_{kg} \cup \{a_i | a_i \in A_w\}$. Our task is to classify each answer as either *"supporting"* or *"not supporting"*. To this end, we construct an evidence triple t_e by replacing the object part of the triple with the short answer retrieved. Therefore, for each $a \in A$ and triple $t = (S; P; O)$, the corresponding evidence triple is $t_e = (S; P; a)$. If the claim triple $t = (S; P; Pid; P\_attr; O)$

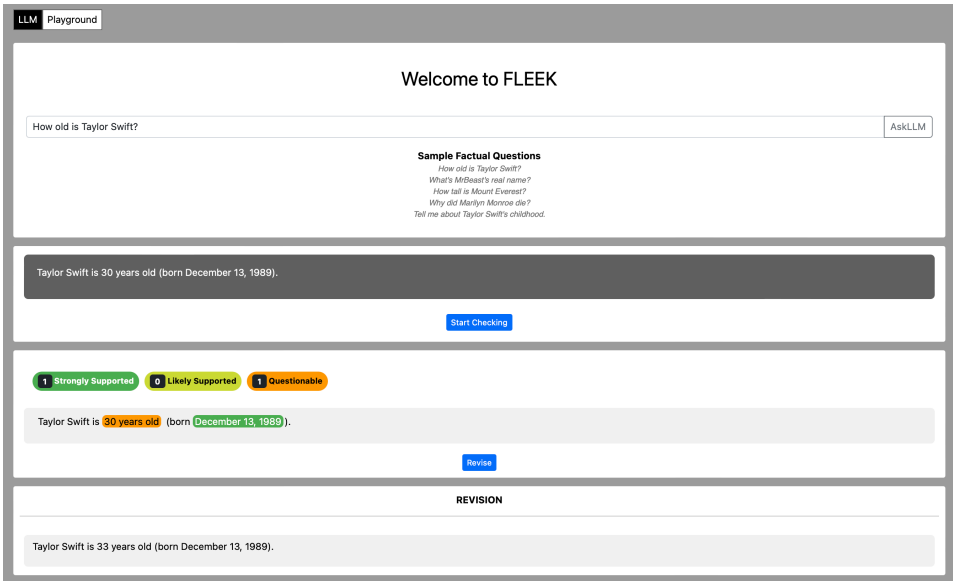


Figure 3: FLEEK LLM view.

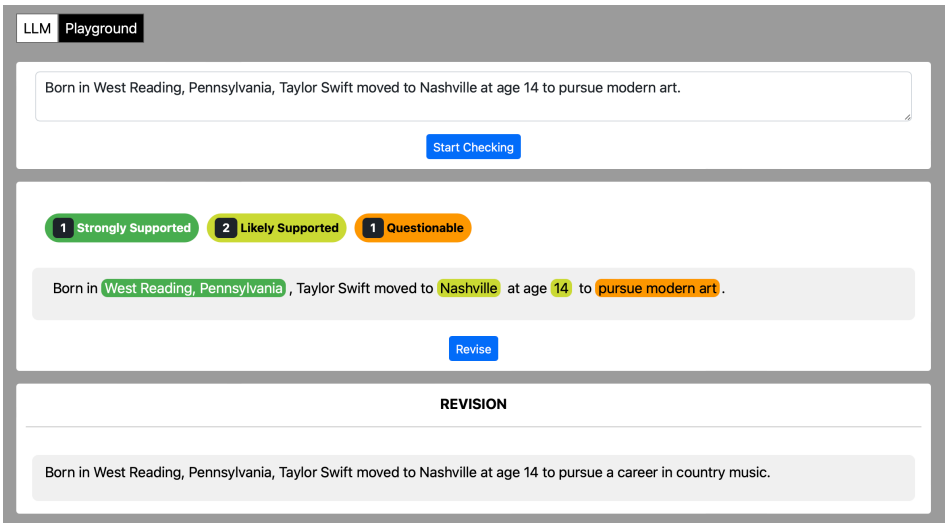


Figure 4: Playground view.

is extended, the corresponding evidence triple is $t_e = (S; P; Pid; P\_attr; a)$. The claim and its corresponding evidence triple are then used to form a prompt and fed to LLM to make a final decision.

2.2 Fact Revision

The Fact Revision module aims to correct a questionable fact triple stated in an input sentence into its corrected version while preserving everything else stated in the sentence. More specifically, let s be a sentence containing a questionable triple t_{src}, i.e., $s \models t_{src}$ (i.e., s entails fact t_{src}). Let the evidence triple formed by the verification process outlined above be t_{dest}. The Fact Revision model will thus rewrite s into s' such that $s' \not\models t_{src} \wedge s' \models t_{dest}$, and $s' \models t_i$ where $t_i \neq t_{src}$ is any triple entailed by s. Following is an example (the objects of the triples are in bold):
s = *"Taylor Swift is **30** years old."*
t_{src} =*(Taylor Swift; age; 30)*
t_{dest} =*(Taylor Swift; age; 33)*
s' = *"Taylor Swift is **33** years old."*
In our implementation, we prompt LLMs with one demonstration to obtain satisfactory results.

3 The User Interface of FLEEK

The frontend of FLEEK is built using Angular[3] and Bootstrap UI[4], which allows for creating dynamic, interactive, and visually appealing user interface. The backend of FLEEK is handled by Django[5], a Python-based server-side framework

[3] https://angular.io/
[4] https://getbootstrap.com/
[5] https://www.djangoproject.com/

that facilitates the integration with ML-based libraries. The entry point to the system is the two views, `LLM` and `Playground`, shown at Figure 3, which we describe next.

LLM View. In this view, the user can check the factual consistency of an LLM (e.g. GPT-3.5) that the user provided (as an endpoint). To interact with FLEEK, first in the **Input Panel** (Figure 3, upper panel), the user can type their query in the question bar, e.g., '`How old is Taylor Swift?`' (or click one of the sample queries) and hit the `AskLLM` button. FLEEK would send the query to the LLM (GPT-3 in this example) and render its response in the **Response Panel** (Figure 3, the upper second panel in dark grey). The verification process will kick in once the user hit the `Start Checking` button. FLEEK verifies the claim(s) made by the LLM by going through the process described in Section 2.1. The verification results are shown in the **Verification Panel** (Figure 3, middle panel). With our design, FLEEK is able to highlight the sub-claims in the original text with different color codes to indicate their factual accuracy categories based on the collected evidence. Additionally, the highlighted spans are clickable, which leads to a detailed dialog containing the evidence associated with the claims (illustrated in Figure 1 and Figure **??**). Evidence retrieved from web are accompanied with a `source link` as well. At the bottom, the user can request FLEEK (hit the `Revise` button) to revise the original claims using the collected evidence. Based on the evidence retrieved from the KG and web, we can have multiple revision alternatives. Verification results for the example shown in Figure 3 and 4 allow for only one possible revision.

Playground View. This view allows the user to verify any specific piece of text of their choosing. This feature empowers users to automatically fact-check tweets, trending news, arbitrary LLM outputs, or even their own writing with just a few clicks. Figure 4 illustrates the view. The user can input their desired text into the designated input panel (scratchpad) and hit the *"Start Checking"* Button (Figure 4, upper panel). The verification and revision process is the same as in the *LLM View*.

4 Evaluation

Previous benchmarks on fact verification (Thorne et al., 2018; Aly et al., 2021) provide a single decision for the entire claim based on the retrieved evidence. However, in this work, we introduce fine-grained fact verification with attribution to external knowledge. As this is the first study on this task, there exist no benchmarks for evaluating FLEEK's performance. Next, we conduct preliminary experiments using manually created evaluation data.

4.1 Evaluation Data Creation

Our system has two use cases. The first one is to verify the responses generated by LLMs (in this case, GPT-3). To evaluate our system's performance, we selected 50 questions from WikiQA (Wikipedia open-domain Question Answering) test set (Yang et al., 2015) and collected their corresponding GPT-3 responses. We then manually annotated each response using the following steps: (1) identify the facts within the response, (2) label each fact as *"Strongly Supported"*, *"Likely Supported"*, or *"Questionable"*, (3) accompany each fact with an evidence set, particularly the questionable facts. We call this dataset $Bench_{LLM}$. Each instance in the $Bench_{LLM}$ contains the annotated GPT-3-generated response.

The second use case is to verify an arbitrary input text. To create evaluation data that suits this task, we target the introduction section of Wikipedia pages. To partially perturb sentences and create incorrect facts, we sample 50 random sentences with at least one hyperlink. Then, we retrieve the hyperlink's corresponding entity from Wikidata [6], find the entity's type (*instance of* property), and retrieve candidate entities with the same type. Finally, we perturb the sentence by replacing the original hyperlink with one randomly selected entity within the candidate list. After perturbation, we annotate the sentence the same way that we created $Bench_{LLM}$. We call this dataset $Bench_{Text}$.

4.2 Large Language Models

All FLEEK's components that facilitate fact verification and correction use few-shot prompting with a large language model. Any model that can learn from in-context demonstrations can be used to instantiate FLEEK. We choose one open-source model, Vicuna (33 billion parameters), and one closed source model, GPT-3 (175 billion parameters), to create two instances of our tool. We call the instance with Vicuna as its large language model FLEEK $_{Vicuna}$ and the instance that utilizes GPT-3 as its large language model FLEEK $_{GPT-3}$. We evaluate both instances in the following section.

[6] https://wikidata.org/

Instance	Category	$Bench_{LLM}$			$Bench_{Text}$		
		P	R	F1	P	R	F1
$FLEEK_{Vicuna}$	Strongly Supported	91.66	84.61	87.99	84.61	91.66	87.99
	Likely Supported	94.73	58.69	72.47	87.5	37.33	52.33
	Questionable	54.54	60	57.13	93.61	74.57	83.01
	Total	88.75	61.73	**72.81**	90.21	56.84	**69.73**
$FLEEK_{GPT-3}$	Strongly Supported	100	95.23	97.55	100	100	100
	Likely Supported	93.22	61.79	74.31	95.77	79.06	86.61
	Questionable	66.66	100	79.99	87.75	69.35	77.47
	Total	89.0	76.06	**82.02**	93.28	77.16	**84.45**

Table 1: Evaluating two instances of FLEEK on Bench_{LLM} and Bench_{Text}.

4.3 Experimental Results

Consider the set of system-generated spans $S = \{s_1, ..., s_n\}$ and ground truth spans $G = \{g_1, ..., g_m\}$. We measure the number of textual spans that are correctly identified, labeled, and attributed to the valid supporting evidence as ov. Then, we calculate verification system's precision as $\frac{ov}{|S|}$, recall as $\frac{ov}{|G|}$, and the F1 score. Table 1 shows the performance of our verification systems on both evaluation datasets. As illustrated, FLEEK $_{GPT-3}$ outperforms FLEEK $_{Vicuna}$ on both datasets by a large margin (12 F1 pts). However, given that Vicuna is about $5\times$ smaller than GPT-3, the average performance of FLEEK $_{Vicuna}$ (71.27 F1) shows its efficiency in Fact Verification. Moreover, the results show that both systems can identify the *"Strongly Supported"* facts with high precision and recall. However, they fail to detect all facts or attribute them to the correct evidence for *"Likely Supported"* or *"Questionable"* cases.

We also measure the accuracy of revisions proposed by the fact correction component. Both systems have on-par performance with an average accuracy of 72.7%. However, our investigation shows that 54.1% of incorrect revisions are a result of errors in previous components propagated through the system. Thus, Fact Correction's average precision, given the correct verification results, is 87.5%.

Note that although our initial results show great promise, both evaluation datasets are small (50 sentences) and come from the same data source (Wikpedia). One ongoing work is to create a larger benchmark (with different levels of difficulty from more diverse sources) for a more extensive and reliable evaluation of our system.

Error Analysis. We randomly select 30 examples where FLEEK $_{GPT-3}$ made erroneous decisions and investigate the types of errors each of its components made (Figure 5). In general, the Fact Extraction component accounted for a significant portion, approximately 49%, of the total errors. This emphasizes the difficulty of mastering

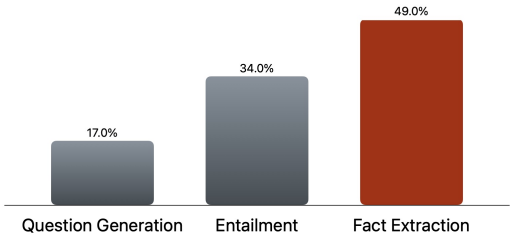


Figure 5: Percentage of total errors generated by different components of FLEEK $_{GPT-3}$.

Fact Extraction through in-context learning. Errors produced by this component include, but are not limited to, wrong triple format, broken n-ary relations, missing triples, and hallucination. Figure 5 further indicates that the GPT-3 might not excel in reasoning, as the entailment component also contributes significantly to system errors.

5 Conclusion and Future Work

We presented FLEEK, an innovative solution geared towards assisting users in verifying the accuracy and factuality of textual claims. We aim to keep improving the FLEEK so that it can be a handy tool for various stakeholders. As part of our future work, we intend to do more comprehensive evaluations of FLEEK, including testing it with various LLMs and over a comprehensive benchmark.

Limitation. First, our current system depends on the initial set of responses generated by LLMs to perform the tasks. Nevertheless, we can prompt each component multiple times and employ methods such as majority voting to enhance the accuracy of each task. Second, experiments presented are based on small-scale datasets. We plan to expand both datasets as part of our future endeavors. Finally, both datasets are manually annotated by one annotator. We plan to hire more annotators and refine the annotation process so as to provide a more comprehensive evaluation of our method.

References

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. FEVEROUS: Fact extraction and VERification over unstructured and structured information.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 601–610, New York, NY, USA. Association for Computing Machinery.

Ihab F. Ilyas, Theodoros Rekatsinas, Vishnu Konda, Jeffrey Pound, Xiaoguang Qi, and Mohamed Soliman. 2022. Saga: A platform for continuous construction and serving of knowledge at scale. In *Proceedings of the 2022 International Conference on Management of Data*. ACM.

Luca Di Liello, Siddhant Garg, Luca Soldaini, and Alessandro Moschitti. 2022. Paragraph-based transformer pre-training for multi-sentence inference.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2020. Fine-grained fact verification with kernel graph attention network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7342–7351, Online. Association for Computational Linguistics.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap.

Ciyuan Peng, Feng Xia, Mehdi Naseriparsa, and Francesco Osborne. 2023. Knowledge graphs: Opportunities and challenges.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HLT*.

Helena Webb, Marina Jirotka, Bernd Carsten Stahl, William Housley, Adam Edwards, Matthew Williams, Rob Procter, Omer Rana, and Pete Burnap. 2016. Digital wildfires: Hyper-connectivity, havoc and a global ethos to govern social media. *SIGCAS Comput. Soc.*, 45(3):193–201.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2019. Reasoning over semantic-level graph for fact checking. *CoRR*, abs/1909.03745.