Multi-hop Evidence Pursuit Meets the Web: Team Papelo at FEVER 2024

Christopher Malon
NEC Laboratories America
Princeton, NJ 08540
malon@nec-labs.com

Abstract

Separating disinformation from fact on the web has long challenged both the search and the reasoning powers of humans. We show that the reasoning power of large language models (LLMs) and the retrieval power of modern search engines can be combined to automate this process and explainably verify claims. We integrate LLMs and search under a *multi-hop evidence pursuit* strategy. This strategy generates an initial question based on an input claim using a sequence to sequence model, searches and formulates an answer to the question, and iteratively generates follow-up questions to pursue the evidence that is missing using an LLM. We demonstrate our system on the FEVER 2024 (AVeriTeC) shared task. Compared to a strategy of generating all the questions at once, our method obtains .045 higher label accuracy and .155 higher AVeriTeC score (evaluating the adequacy of the evidence). Through ablations, we show the importance of various design choices, such as the question generation method, medium-sized context, reasoning with one document at a time, adding metadata, paraphrasing, reducing the problem to two classes, and reconsidering the final verdict. Our submitted system achieves .510 AVeriTeC score on the dev set and .477 AVeriTec score on the test set.

1 Introduction

Since 2018, the FEVER shared task has challenged natural language processing systems to verify claims using a corpus and provide evidence that witnesses these verdicts. It has evolved from a simple combination of natural language inference (NLI) and entailment (Thorne et al., 2018) to a challenge involving adversarially constructed claims (Thorne et al., 2019), to a challenge to verify complex, multi-hop claims using a combination of tables and free text (Aly et al., 2021). In the current task, it finally arrives at combating real-life disinformation on the web (Schlichtkrull et al., 2023).

Systems are challenged to classify claim texts as supported, refuted, not enough evidence, or conflicting evidence/cherrypicking. In addition to classifying the claim, the systems must submit a list of questions and answers about a claim as evidence, with each answer derived from information on the open web and cited with a URL. Credit is given only when both the classification matches the ground truth and the evidence is adequate. The AVeriTeC score determines evidence adequacy by thresholding an average of METEOR scores between each gold QA pair and the corresponding submitted QA pair in the best assignment of QA pairs.

This task may involve retrieval and reasoning skills at a level for which professional journalists are sometimes employed. The reasoning may involve quote verification, stance detection, or numerical comparisons. The retrieval challenge goes beyond previous political fact-checking tasks (Ostrowski et al., 2021; Alhindi et al., 2018) and even beyond previous FEVER tasks in advancing from a closed corpus (Wikipedia) to the open web.

Whereas previous FEVER shared tasks needed to be solved by researcher-trained models, the current shared task allows the use of commercial API components. The winning team in FEVEROUS based their retriever on fitting a Dense Passage Retriever (Karpukhin et al., 2020) to the FEVEROUS data (Bouziane et al., 2021), but the training data for FEVER 2024 is quite limited, consisting of only 3,068 claims, and a retriever trained on user feedback from worldwide search queries should easily be more powerful. Additionally, an external web search engine such as Google Search may provide additional query understanding features not found in DPR, as a recent feature (not in the API

Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER), pages 37–46

we used) applies generative AI to search[1]. Even though the gold evidence documents are guaranteed to appear in the knowledge store provided by the contest organizers, the snippets may not be extracted successfully. We found that 297 of the 500 claims in the dev set included gold documents with empty extracted text. In contrast, web search provides at least some text even from pages that the provided web scraper is blocked from accessing. Therefore, we chose to incorporate web search into our system.

Relying on a large language model (LLM) such as GPT-4o (OpenAI, 2024) for reasoning lets us leverage skills that could not be learned from 3,068 heterogenous claims, and go beyond the simple semantic comparison of an NLI model. Beyond simple NLI, ChatGPT and GPT-4 have been utilized to detect hallucinations in text summaries (Luo et al., 2023), as multi-faceted evaluators that score generated text (Zheng et al., 2023), and for critiques and corrections of generated text (Lin et al., 2024).

Though there are many ways of using a search engine and LLM within a fact-checking system, our main contribution is to show the power of combining them in a strategy of *multi-hop evidence pursuit*, which formulates additional questions only after searching and formulating answers to previous questions. In the following sections, we also investigate the impact of many choices of how the questions could be generated, the nature and size of context for generating answers, handling of multiple search results, metadata, paraphrasing, reducing the problem to two classes, and reconsidering the final verdict.

2 Related work

Retrieval-augmented generation (RAG) (Lewis et al., 2020) provides a general paradigm for enabling an LLM to answer questions that surpass the knowledge encoded in the LLM parameters, which is a task somewhat isomorphic to verifying claims (Demszky et al., 2018).

A growing body of work utilizes LLMs as high-level reasoning controllers that can solve tasks by querying agents to provide information or solve subproblems (Xi et al., 2023; Wu et al., 2023a). An early example for fact-checking an LLM's own output was LLM-Augmenter (Peng et al., 2023), which called an open retrieval pipeline as an agent action to iteratively improve an LLM response. Chan et al. (2024) uses an LLM to rewrite, decompose, and disambiguate queries before searching, and these steps are made into a hierarchy of agents in Chen et al. (2024). Wang et al. (2024) used a combination of Google search and GPT-4 with a single hop to fact-check claims in the FacTool-KB, FELM-WK, and HaluEval datasets. Behind a closed API, SearchGPT has been launched in beta to a few users as a service to provide access to a search-empowered OpenAI LLM.[2]

FEVER 2024 presents a multi-hop, open corpus fact verification challenge. In the multi-hop shared task of FEVEROUS, all but two contestants collected all the needed evidence up front, after only reading the claim (Aly et al., 2021). Later top performers (DCUF, UniFee, SEE-ST) addressed evidence interaction with graph-based methods but still did not address evidence that might be missed by the initial document retrieval (Hu et al., 2022, 2023; Wu et al., 2023b). Malon (2021) established an iterative paradigm for fact verification that retrieves further documents, sentences, and table cells by generating follow-up queries that are formulated after considering only the first retrieval, which we follow in the present system, in *multi-hop evidence pursuit*.

In medical question answering, Xiong et al. (2024) contemporaneously has proposed "iterative RAG for medicine" which uses an LLM to generate follow-up questions considering previous retrievals. In our algorithm, the relevance of each question is assured by generating it only upon a failure to verify the claim as true or false based on the existing evidence. Their method may generate irrelevant questions after an answer could already be obtained, simply because the fixed numbers of questions are not achieved, resulting in lower evidence relevance and higher computational cost. Our system can stop as soon as a verdict is clear, and if our system is configured to generate additional questions by paraphrasing, their relevance is assured by their similarity to the original questions.

3 Methodology

3.1 Overall architecture

Pseudocode outlining the overall system is given in Algorithm 1, with the main loop shown in Figure 1. At the core of the system are question generation functions $GetFirstQuestion$ and $GetNextQuestion$, for which we consider

[1] https://blog.google/products/search

[2] openai.com/index/searchgpt-prototype/

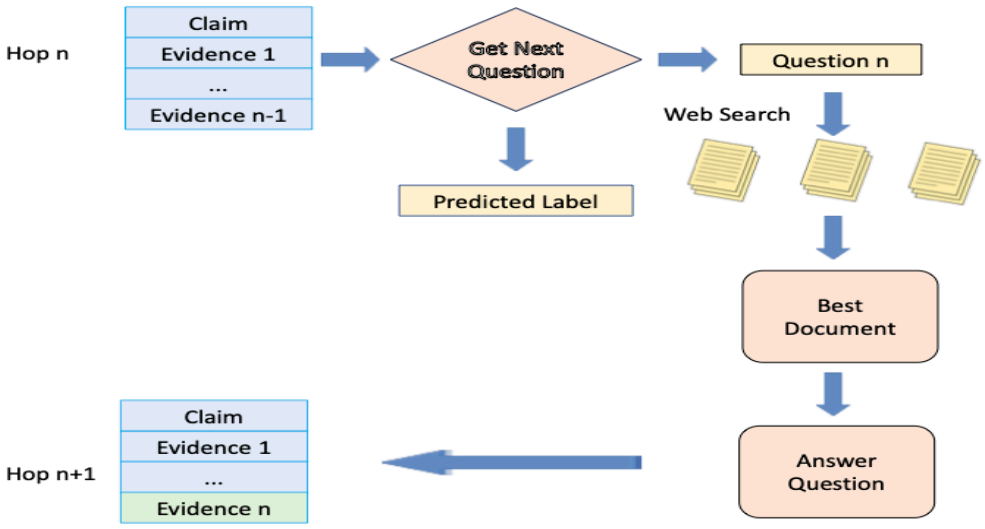

Figure 1: Pursuing additional evidence by generating follow-up questions.

implementations either by sequence-to-sequence encoder-decoder transformers such as T5 (Raffel et al., 2020), or by an LLM. The $GetAnswer$ function (Algorithm 2) prompts an LLM to implement $LLMBestDoc$ and $LLMAnswer$ to answer the generated questions. The final verdict is also chosen by prompting an LLM with the generated questions and answers, in $LLMVerdict$.

Algorithm 1. Claim verification

Input: Claim c, max questions n
Initialize QA list $Q = \emptyset$
Let $q = GetFirstQuestion(c)$
while $|Q| < n$ and $q \neq True$ and $q \neq False$
  Let $a = GetAnswer(q, c)$
  Append (q, a) to Q
  Let $q = GetNextQuestion(c, Q)$
  # $GetNextQuestion$ outputs True or False
  # if next question not needed
Let $k = |Q|$
while $|Q| < n$
  Let $i = |Q|$
  Let $q = Paraphrase(q_{i \bmod k})$
  Let $a = GetAnswer(q, c)$
  Append (q, a) to Q
Output: $v = LLMVerify(Q, c)$ and Q

Unlike the baseline system (Schlichtkrull et al., 2023), our system does not generate questions on a *post hoc* basis after finding evidence, but generates questions before web searches, playing a key role in steering the verification process. Rather than assuming all evidence can be found up front with a single search query, we review the current set of evidence and generate text (in our case, a question) that provides a query to search for what is still missing and needed after each hop, like the followup queries introduced in Malon (2021). Whereas the queries in Malon (2021) were generated by training a sequence to sequence model to predict what the missing evidence would look like, our system prompts an LLM to ask a question that the missing evidence answers.

Algorithm 2. Function $GetAnswer(q, c)$

Input: Question q, claim c
Let $s = c + q$ concatenation
Let $G = WebSearch(s)$
if $G = \emptyset$:
  Let $G = WebSearch(NamedEntities(s))$
$G = \{(url_0, quote_0), \ldots, (url_9, quote_9)\}$
Let $i = LLMBestDoc(G, q)$
Let $d = FullDocument(url_i)$
Let $e = AlignContext(d, quote_i, 5)$
Output: $a = LLMAnswer(q, e)$

The generation of evidence QA pairs temporarily stops when $GetNextQuestion$ thinks it can classify the claim as supported or refuted without asking another followup question (see Appendix B). After that point, the already generated questions are paraphrased using an LLM and corresponding answers are found until the desired number of QA pairs is obtained. Finally, an LLM uses all QA

pairs to decide the final classification for the claim.

3.2 Question generation

We consider two variants for the functions $GetFirstQuestion$ and $GetNextQuestion$. In the **Seq** version, we finetune a sequence-to-sequence encoder-decoder transformer model. For $GetFirstQuestion$, the input is the claim, and the output is the first question. For $GetNextQuestion$, the input is the claim concatenated with all previous question-answer pairs, in the format

Claim: *claim* Question: $question_0$
Answer: $answer_0$ Question: $question_1$
Answer: $answer_1 \ldots$

and the output is the next question to be generated. These input strings are prefixed with the string "question: ". Details of the fine-tuning procedure are in Appendix A. Question-answer pairs from the gold data in the training set are used for this fine-tuning.

The other variant is the **LLM** version, in which we prompt the LLM with similar inputs. The prompts are given in Appendix B. Because LLM output is often verbose and may contain unnecessary explanations, we sentence split the output and use only the first sentence containing a question mark. If this is impossible, we use the whole output.

If an adequate number of questions and answers has been generated and the verdict is clear, the model has the opportunity to output a *True* or *False* verdict to stop the question generation.

As a further ablation, we consider a more traditional technique of generating all the questions at once, given the claim. The function $AllAtOnce$ (prompt in Appendix B) replaces $GetFirstQuestion$ to generate a set of questions, and the **while** loop in Algorithm 1 is replaced by a loop over the generated questions, calling $GetAnswer$ but not $GetNextQuestion$.

3.3 Evidence selection

Here we describe the function $GetAnswer$, displayed in Algorithm 2, which retrieves evidence and uses it to answer the generated questions. Prompts for its LLM helper functions are given in Appendix B.

The generated question is concatenated to the claim to form a web search query, and the top ten search results are obtained, including their URL, the short snippet displayed in the search results, and usually the page title, site name, and publication date. When the web search returns no results, we retry the search using only the named entities (and other capitalized words after the first word) from the initial search query, following the supplemental queries which improved retrieval by Wikipedia page title lookups in Malon (2018).

By prompting, $LLMBestDoc$ is used to choose one document that best answers the question from the set of ten web search hits. We attempt to retrieve and scrape the text of that document using its URL (function $FullDocument$). This is implemented using the `scrape_text_from_url` function provided in the AVeriTeC baseline (Schlichtkrull et al., 2023), which uses the Python *trafilatura* library.[3] If the scraping succeeds, we look for a small window of text (five sentences in our experiments) that best overlaps the web search snippet (function $AlignContext$). Specifically, all five-sentence windows of the document that include more than 70% of the words in the web search snippet are recorded in order, and the middle such window is taken. Using this window as the document excerpt provides more background and context to the text that web search found to be relevant, while avoiding prompting with the overwhelming amount of text that might be found in the full web page. If the scraping fails, we continue to the next stage using only the web search snippet as document text.

Because $LLMBestDoc$ depends on parsing LLM output, it may fail to choose a best document. If a best document is chosen and the scraping succeeds, the LLM is prompted to answer the question using the selected five-sentence window of the best document in $LLMAnswer$. If the best document is chosen and the scraping fails, $LLMAnswer$ is run using the text of the web search snippet only. If a best document was not chosen in $LLMBestDoc$, we use the full text of the LLM response in that function as the answer and the web search result page itself as the evidence.

In $LLMBestDoc$ and $LLMAnswer$, the prompt includes not only the text for each document, but metadata including the page title, site name, and publication date, when this metadata appears in web search results. This metadata may occasionally be useful in assessing the credibility or relevance of the information to the question.

[3] `github.com/adbar/trafilatura`

3.4 Reconsideration and Classification

The $Paraphrase$ function asks the LLM for paraphrases of the existing questions. In practice, multiple paraphrases of each question are requested at once to avoid repeated calls, even though they are used one at a time. Although these paraphrases may not be logically necessary once $GetNextQuestion$ has determined a verdict, sometimes they provide a chance to reconsider the same questions using multiple sources. The variations in wording also improve the AVeriTec score, as discussed in section 4.

The $LLMVerdict$ function is called after all question-answer pairs are collected, to choose the predicted label for each example. Using additional QA pairs, it may override the decision that stopped the QA generation process. Table 1 shows the distribution of labels in the training and development sets. "Not Enough Evidence" and "Conflicting evidence / cherrypicking" are minority classes, and we were unable to predict them with good F1 score. We obtained a higher score by limiting $LLMVerdict$ to predicting "Supports" or "Refutes."

Class	Train	Dev
Supported	27.7%	24.4%
Refuted	56.8%	61.0%
NEI	9.2%	7.0%
Conflicting	6.4%	7.6%

Table 1: Distribution of class labels.

4 Experiments

We implement Algorithm 1 using GPT-4o (`gpt-4o-2024-05-13`, seed 42) as the LLM, T5 (`t5-large`) (Raffel et al., 2020) as the sequence-to-sequence model, and Google as the web search engine, and consider various ablations. For a faster development cycle and reduced monetary cost, Table 2 reports the performance of each of our systems only on the first 200 examples of the development set.

4.1 Question formation

Recall from Section 3.2 that in Algorithm 1, the functions $GetFirstQuestion$ and $GetNextQuestion$ could be implemented either by **Seq** or **LLM**, or instead of Algorithm 1, the questions could be generated $AllAtOnce$. Whichever question generation approach is used, at most five questions are taken from the question generator and the paraphrase loop of Algorithm 1 extends the list to five questions. The submitted system follows Algorithm 1 using **Seq** for $GetFirstQuestion$, and **LLM** for $GetNextQuestion$ (Seq+LLM).

The lower performance of the $AllAtOnce$ alternative indicates that this task requires followup searches considering the evidence already retrieved, with searches that cannot be anticipated using the claim alone. It validates our choice to use a *multi-hop evidence pursuit* strategy (Malon, 2021).

The LLM+LLM alternative shows that performance worsens if we generate the first question using GPT-4o. An inspection of the data revealed that the gold first questions were usually simple rephrasings of the claims, which T5 can learn well, whereas GPT-4o often tried to generate something more complicated.

The Seq+Seq alternative shows that performance worsens if we generate the subsequent questions using T5. Subsequent gold questions often reflected deeper reasoning using the obtained answers, which we suspect are beyond the capabilities of simple sequence to sequence models.

4.2 Label prediction

We have implementations of $LLMVerdict$ that use a four-class prompt, or eliminate the "Not Enough Evidence" (NEI) and "Conflicting Evidence / Cherrypicking" classes to decide only between "Supported" and "Refuted." The 4-class result (otherwise the same as the main system) shows very low F1 scores for the NEI and Conflicting classes. As NEI claims form only 7.0% of the dev set and Conflicting claims form only 7.6%, we decided that it is always best to guess another label.

Another variant, "No late verdict," calls $LLMVerdict$ only if the **while** loop is not terminated by predicting True or False, and maintains that early decision even after the paraphrases are added. (If True is obtained, "Supported" is predicted and if False is obtained, "Refuted" is predicted.) The difference in label accuracy shows it is sometimes useful to consider the whole question and answer chain from the beginning when forming a verdict.

4.3 Answer formation

The submitted system uses $FullDocument$ and $AlignContext$ to obtain longer document contexts

System	Supp F1	Ref F1	NEI F1	Conf F1	Acc	AVeriTec 0.25
AllAtOnce	.591	.813	0	0	.705	.340
LLM+LLM	.644	.821	0	0	.720	.385
Seq+Seq	.638	.816	0	0	.715	.370
4 class	.486	.593	**.148**	**.069**	.415	.245
No late verdict	.643	.811	0	0	.705	.450
No long doc	.577	.819	0	0	.705	.465
Multi-doc	.673	.837	0	0	.735	.460
No metadata	.575	.810	0	0	.700	.410
No paraphrase	.701	.839	0	0	.745	.225
Repeat not para	.624	.813	0	0	.710	.340
Algorithm 1	**.716**	**.841**	0	0	**.750**	**.495**

Table 2: Results on the first 200 examples of the dev set

Data	Submission	Supp F1	Ref F1	NEI F1	Conf F1	Acc	AVeriTec 0.25
Dev	Algorithm 1	.698	.853	0	0	.754	.486
Dev	Inflated to 10	.698	.853	0	0	.754	.510
Test	Algorithm 1	—	—	—	—	—	.445
Test	Inflated to 10	—	—	—	—	—	.477

Table 3: Final results on full datasets

for prompting $LLMAnswer$. The "No long doc" ablation uses only the original web search snippet as context for $LLMAnswer$. The close performance in AVeriTeC score shows that while longer context is helpful, it is often unnecessary. Scraping web pages to obtain this longer context has become difficult as many sites seek to restrain robots, so relying on snippets is convenient. In cases where our scraping fails, the original snippet is returned by $FullDocument$ anyway.

The "Multi-doc" ablation calls $LLMAnswer$ using all ten search hits and their snippets, without calling $LLMBestDoc$ to focus on one. It is very close to our system in label accuracy. Although it narrows the depth and context of information presented to $LLMAnswer$, it may have advantages in presenting multiple possible perspectives.

Metadata for each document context is usually presented to $LLMAnswer$ in the form

$$\text{Document } i : (title, \text{from } site, \text{published } date)$$

The lower label accuracy and AVeriTeC score of the "No metadata" variant show that knowing where evidence came from is helpful to the LLM.

4.4 Evidence length

When the label is predicted correctly for an example, the AVeriTeC score thresholds an example score, which is computed as the sum of the METEOR scores between gold QA pairs and best matching predicted QA pairs, divided by the number of gold QA pairs. Whenever fewer QA pairs are predicted than gold QA pairs, those gold QA pairs contribute zero to this average. Therefore, to optimize the AVeriTec score, it is important to predict at least as many QA pairs as the number of gold pairs, even if the some predicted pairs match poorly.

A system could submit up to ten QA pairs for each example. However, only 5% of examples had more than five gold QA pairs in the development set. Since the ultimate objective is optimizing human evaluation rather than AVeriTeC score and reading more than five QA pairs may be frustrating for a human, we initially applied our systems to produce five QA pairs per question.

For many examples, Algorithm 1 could reach decisions of $q = True$ or $q = False$ in its first loop of $GetFirstQuestion$ and $GetNextQuestion$ using fewer than five QA pairs. We compared the score obtained by *repeating* QA pairs, or by asking the LLM to *paraphrase* the existing questions in the second loop of Algorithm 1, until five QA pairs were obtained. In the case of *paraphrase*, new answers are sought for the rewritten questions. Besides improving the AVeriTeC score, the new an-

swers may be used to reconsider the final verdict.

The "No paraphrase" ablation has a minimal effect on label accuracy, but since fewer QA pairs are generated, AVeriTec score is less than half the score of the submitted system. "Repeat not paraphrase" to get five QA pairs can recover some of the AVeriTeC score, but the paraphrases help the METEOR score of the best assignment much more than duplicates.

Ten QA pairs is the upper limit, and submitting additional QA pairs up to ten can only improve the score of the best assignment between submitted pairs and gold pairs. We took our five generated QA pairs from Algorithm 1 ($GetFirstQuestion$, $GetNextQuestion$, and paraphrasing) and duplicated them to submit ten. Naturally, repeating can be helpful if one generated QA pair addresses points raised in multiple gold QA pairs. The effect of inflating the QA pairs on our full dev set and test set performance is shown in Table 3.

5 Conclusion

The AVeriTeC shared task is a realistic fact-checking challenge on actual web disinformation. The best large language models offer the deep reasoning power needed to pursue missing evidence to verify claims, and the best web search engines provide the vast document indices and retrieval capabilities needed to find it.

We have contributed a multi-hop evidence pursuit framework which combines the strengths of sequence to sequence models with LLMs to generate first question and subsequent questions separately, considering the present information; to stop pursuit once the answer is clear; and to embellish evidence by paraphrasing before considering the whole evidence chain to make the final verdict. Ablations indicate the importance of each design choice. Multi-hop evidence pursuit outperforms trying to generate all questions in one step. Reducing the number of classes, and using metadata and multi-sentence context from one best document, were important in obtaining our best performance.

The fact checking system presented may be useful to expedite the work of human fact checkers or provide a more rapid preliminary response to disinformation. Its full explainability could mitigate the effect of misclassifications, if the explanations were read and considered by a human. Over a history of many claims, ratings of disinformation from our system and/or human fact checkers could be used to rate the credibility of an information source.

Limitations

When "Not Enough Evidence" (NEI) is an option, an LLM tends to select it too often. Our system was unable to predict either NEI or "Conflicting Evidence / Cherrypicking" with acceptable accuracy. Considering this, and the fact the overall label accuracy is only .754, humans should be cautious in trusting this system's output to verify a claim without reading the rationale.

LLMs have insufficient information to judge the overall credibility of a website, and currently just the site name is given for the LLM's consideration. Metadata including the site name helps (to give an example from the dev set, GPT-4o was aware or discovered through its searches that Scoopertino was a satirical website), but generally, misinformation that is corroborated elsewhere on the web may fool our fact checking system.

Although the LLM is always prompted to answer questions "based on the above information" quoted from retrieved documents or its previous answers, there is no guarantee that the LLM does not apply other, untraceable knowledge in forming its answers. We use a date filter to ensure that all web searches return documents only from before each claim date, but we use an LLM whose training cutoff is after the claim dates.

Novel information first reported, which has no basis in existing documents, can never be fact-checked with the techniques of this system (for example, the first report that a presidential candidate was shot). That kind of fact checking requires judgments of plausibility, credibility, and consistency that are out of scope for this system.

References

Tariq Alhindi, Savvas Petridis, and Smaranda Muresan. 2018. Where is your evidence: Improving fact-checking by justification modeling. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 85–90, Brussels, Belgium. Association for Computational Linguistics.

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. The fact extraction and VERification over unstructured and structured information (FEVEROUS) shared task. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 1–13, Dominican Republic. Association for Computational Linguistics.

Mostafa Bouziane, Hugo Perrin, Amine Sadeq, Thanh Nguyen, Aurélien Cluzeau, and Julien Mardas. 2021. FaBULOUS: Fact-checking based on understanding of language over unstructured and structured information. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 31–39, Dominican Republic. Association for Computational Linguistics.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. Rq-rag: Learning to refine queries for retrieval augmented generation. *Preprint*, arXiv:2404.00610.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Jiangning Liu, Wenwei Zhang, Kai Chen, and Feng Zhao. 2024. Mindsearch: Mimicking human minds elicits deep ai searcher. *Preprint*, arXiv:2407.20183.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *Preprint*, arXiv:1809.02922.

Nan Hu, Zirui Wu, Yuxuan Lai, Xiao Liu, and Yansong Feng. 2022. Dual-channel evidence fusion for fact verification over texts and tables. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5232–5242, Seattle, United States. Association for Computational Linguistics.

Nan Hu, Zirui Wu, Yuxuan Lai, Chen Zhang, and Yansong Feng. 2023. UnifEE: Unified evidence extraction for fact verification. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1150–1160, Dubrovnik, Croatia. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Zicheng Lin, Zhibin Gou, Tian Liang, Ruilin Luo, Haowei Liu, and Yujiu Yang. 2024. Criticbench: Benchmarking llms for critique-correct reasoning. *Preprint*, arXiv:2402.14809.

Zheheng Luo, Qianqian Xie, and Sophia Ananiadou. 2023. Chatgpt as a factual inconsistency evaluator for text summarization. *Preprint*, arXiv:2303.15621.

Christopher Malon. 2018. Team papelo: Transformer networks at FEVER. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 109–113, Brussels, Belgium. Association for Computational Linguistics.

Christopher Malon. 2021. Team papelo at FEVEROUS: Multi-hop evidence pursuit. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 40–49, Dominican Republic. Association for Computational Linguistics.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Wojciech Ostrowski, Arnav Arora, Pepa Atanasova, and Isabelle Augenstein. 2021. Multi-hop fact checking of political claims. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 3892–3898. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *Preprint*, arXiv:2302.12813.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Michael Schlichtkrull, Zhijiang Guo, and Andreas Vlachos. 2023. Averitec: A dataset for real-world claim verification with evidence from the web. In *Advances in Neural Information Processing Systems*, volume 36, pages 65128–65167. Curran Associates, Inc.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018. The fact extraction and VERification (FEVER) shared task. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2019. The FEVER2.0 shared task. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pages 1–6, Hong Kong, China. Association for Computational Linguistics.

Yuxia Wang, Revanth Gangi Reddy, Zain Muhammad Mujahid, Arnav Arora, Aleksandr Rubashevskii, Jiahui Geng, Osama Mohammed Afzal, Liangming Pan, Nadav Borenstein, Aditya Pillai, Isabelle Augenstein, Iryna Gurevych, and Preslav Nakov. 2024. Factcheck-bench: Fine-grained evaluation benchmark for automatic fact-checkers. *Preprint*, arXiv:2311.09000.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023a. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Zirui Wu, Nan Hu, and Yansong Feng. 2023b. Enhancing structured evidence extraction for fact verification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6631–6641, Singapore. Association for Computational Linguistics.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey. *Preprint*, arXiv:2309.07864.

Guangzhi Xiong, Qiao Jin, Xiao Wang, Minjia Zhang, Zhiyong Lu, and Aidong Zhang. 2024. Improving retrieval-augmented generation in medicine with iterative follow-up questions. *Preprint*, arXiv:2408.00727.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

A Fine-tuning

A `t5-large` model was fine-tuned for three epochs with batch size 4, maximum source length 64 or 256 for $GetFirstQuestion$ or $GetNextQuestion$, and maximum target length 64. For the AdamW optimizer, default Huggingface values of 5×10^{-5} were used for the learning rate, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The model was prompted with the prefix "question: " followed by the inputs. Only gold data from AVeriTeC was used for the fine-tuning of each model.

B Prompts

GetFirstQuestion. For the LLM variant, the prompt is:

> We are trying to verify the following claim by *speaker* on *date*. Claim: *claim* We aren't sure whether this claim is true or false. Please write one or more questions that would help us verify this claim, as a JSON list of strings. Keep the list short.

The JSON is parsed and only the first string in the list is used.

AllAtOnce. For the $AllAtOnce$ variant, we use the same prompt as $GetFirstQuestion$ to get the questions, but we keep the entire list.

GetNextQuestion. For the LLM variant, the prompt is:

> We are trying to verify the following claim by *speaker* on *date*. Claim: *claim* So far we have asked the questions: Question 0: $question_0$ Answer: $answer_0$ Question 1: $question_1$ Answer: $answer_1 \ldots$ Based on these questions and answers, can you verify whether the claim is true or false? Please answer `[[True]]` or `[[False]]`, or ask one more question that would help you verify.

The response is searched for `[[True]]` or `[[False]]`. If neither is found, then the response is sentence tokenized with the `sent_tokenize` function of NLTK 3.8.1 and the first sentence that includes a question mark is returned.

LLMBestDoc. The prompt is:

> We searched the web and found the following information. Document 0 ($title_0$, from $site_0$, published $date_0$): $snippet_0$ Document 1 ($title_1$, from $site_1$, published $date_1$): $snippet_1$ ... Document 9 ($title_9$, from $site_9$, published $date_9$): $snippet_9$ Based on the above information, please answer the following question, referring to the one document that best answers the question. $question$

Note that the original claim is not used in this prompt. The response is searched with a regex for the first instance of `Document\s+([0-9])` / or `Documents[ 0-9,]+and ([0-9]+)` and the corresponding numbered document is taken. If the regex search fails, the search result page itself is used as context for answering the question, and the full response is used as the answer.

LLMAnswer. Unlike $LLMBestDoc$, this is called with context from one document. The prompt is:

> We searched the web and found the following information. Document ($title$, from $site$, published $date$): $context$ Based on the above information, please answer the following question. $question$

The entire response is used as the answer.

Paraphrase. The prompt is:

> Please give four ways to rephrase the following question. Give your answer as a JSON list of strings, each string being one question. Question: $question$

LLMVerify. The prompt is:

> We are trying to verify the following claim: $claim$ Based on our web searches, we resolved the following questions. $0.$ $question_0$ $answer_0$ $\ldots k.$ $question_k$ $answer_k$ Is the claim (A) fully supported by the evidence, or (B) contradicted by the evidence? Please respond in the format `[[A]]` or `[[B]]`.

We search the response for `[[A]]` or `[[B]]`. For the four class variant, the end of the prompt is:

> Is the claim (A) fully supported by the evidence, (B) contradicted by the evidence, (C) insufficient information, or (D) conflicting evidence? Please respond in the format `[[A]]`, `[[B]]`, `[[C]]`, or `[[D]]`.