AIC CTU system at AVeriTeC: Re-framing automated fact-checking as a simple RAG task

Herbert Ullrich
AI Center @ CTU FEE
Charles Square 13
Prague, Czech Republic
ullriher@fel.cvut.cz

Tomáš Mlynář
AI Center @ CTU FEE
Charles Square 13
Prague, Czech Republic
mlynatom@fel.cvut.cz

Jan Drchal
AI Center @ CTU FEE
Charles Square 13
Prague, Czech Republic
drchajan@fel.cvut.cz

Abstract

This paper describes our 3^{rd} place submission in the AVeriTeC shared task in which we attempted to address the challenge of fact-checking with evidence retrieved in the wild using a simple scheme of Retrieval-Augmented Generation (RAG) designed for the task, leveraging the predictive power of Large Language Models. We release our codebase[1], and explain its two modules – the Retriever and the Evidence & Label generator – in detail, justifying their features such as MMR-reranking and Likert-scale confidence estimation. We evaluate our solution on AVeriTeC dev and test set and interpret the results, picking the GPT-4o as the most appropriate model for our pipeline at the time of our publication, with Llama 3.1 70B being a promising open-source alternative. We perform an empirical error analysis to see that faults in our predictions often coincide with noise in the data or ambiguous fact-checks, provoking further research and data augmentation.

1 Introduction

We release a pipeline for fact-checking claims using evidence retrieved from the web consisting of two modules – a *retriever*, which picks the most relevant sources among the available knowledge store[2] and an *evidence & label generator* which generates evidence for the claim using these sources, as well as its veracity label.

Our pipeline is a variant of the popular Retrieval-augmented Generation (RAG) scheme (Lewis et al., 2020), making it easy to re-implement using established frameworks such as Langchain, Haystack, or our attached Python codebase for future research or to use as a baseline.

This paper describes our pipeline and the decisions taken at each module, achieving a simple yet efficient RAG scheme that improves dramatically across the board over the baseline system from (Schlichtkrull et al., 2024), and scores third in the AVeriTeC leaderboard as of August 2024, with an AVeriTeC test set score of 50.4%.

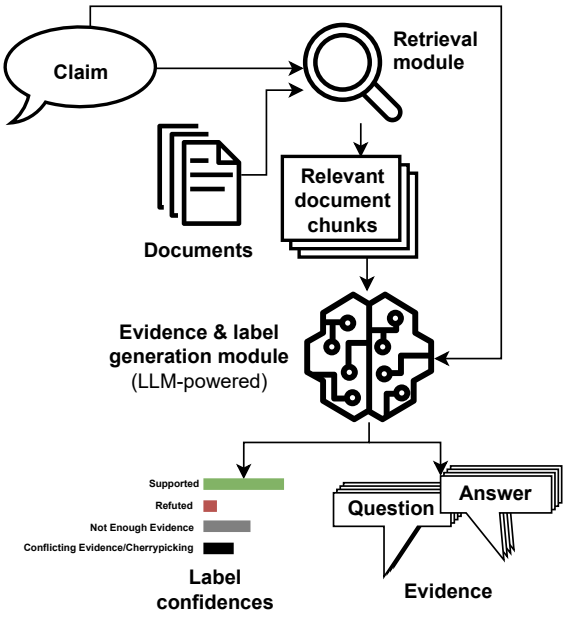

Figure 1: Our pipeline

2 Related work

1. **AVeriTeC shared task** (Schlichtkrull et al., 2024) releases the dataset of real-world fact-checked claims, annotated with evidence available at the date the claim was made.

   It proposes the **AVeriTeC Score** – a method of unsupervised scoring of fact-checking pipeline against this gold data using Hungarian METEOR score, matching the evidence questions (Q) or the whole evidence (Q+A). The score is then calculated as the proportion of claims with accurate label and sound ev-

[1] https://github.com/aic-factcheck/aic_averitec

[2] Due to the pre-retrieval step that was used to generate knowledge stores, our "retriever" module could more conventionally be referred to as a "reranker", which we refrain from, to avoid confusion with reranking steps it uses as a subroutine.

Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER), pages 137–150

idence (using a threshold for Hu-METEOR such as 0.25) among all claims in the dataset, giving an estimate on "how often the whole fact-checking pipeline succeeds end to end".

The provided **baseline** is a pipeline of search query generation, API search (producing a knowledge store), sentence retrieval, Question-and-answer (QA) generation, QA reranking, QA-wise claim classification and label aggregation, achieving an overall AVeriTeC test set score of 11%.

2. **FEVER Shared Task** (Thorne et al., 2018b), a predecessor to the AVeriTeC, worked with a similar dataset engineered on top of the enclosed domain Wikipedic data rather than real-world fact-checks. Its top-ranking solutions used a simpler pipeline of Document Retrieval, Sentence Reranking and Natural Language Inference, improving its modules in a decoupled manner and scoring well above 60% in a similarly computed FEVER score (Thorne et al., 2018a) on this data.

3. **Our previous research** on fact-checking pipelines (Ullrich et al., 2023; Drchal et al., 2023) using data similar to FEVER and AVeriTeC shows significant superiority of fact-checking pipelines that **retrieve the whole documents** for the inference step, rather than retrieving out-of-context sentences.

4. **Retrieval-Augmented Generation (RAG) for Knowledge-Intensive Tasks** (Lewis et al., 2020) takes this a step further, leveraging Large Language Model (LLM) for the task, providing it the whole text of retrieved documents (each a chunk of Wikipedia) and simply instructing it to predict the evidence and label on top of it, achieving results within 4.3% from the FEVER state of the art by the time of its publication (December 2020) *without* engineering a custom pipeline for the task.

3 System description

Our system design prioritizes simplicity, and its core idea is using a straightforward RAG pipeline without engineering extra steps, customizing only the retrieval step and LLM prompting (Listing 1 in Appendix A). Despite that, this section describes and justifies our decisions taken at each step, our additions to the naive version of RAG modules to tune them for the specific task of fact-checking, and their impact on the system performance.

3.1 Retrieval module

To ease comparison with the baseline and other systems designed for the task, our system does not use direct internet/search-engine access for its retrieval, but an AVeriTeC *knowledge store* provided alongside each claim.

To use our pipeline in the wild, our retrieval module is decoupled from the rest of the pipeline and can be swapped out in favour of an internet search module such as SerpApi[3] as a whole, or it can be used on top of a knowledge store emulated using large crawled corpora such as CommonCrawl[4] and a pre-retrieval module.

3.1.1 Knowledge stores

Each claim's knowledge store contains pre-scraped results for various search queries that can be derived from the claim using human annotation or generative models. The knowledge stores used with ours as well as the baseline system can be downloaded from the AVeriTeC dataset page[5], containing about 1000 pre-scraped *documents*[6], each consisting of 28 sentences at median[6], albeit varying wildly between documents. The methods used for generating the knowledge stores are explained in more detail by Schlichtkrull et al. (2024).

Our retrieval module then focuses on picking a set of k ($k = 10$ in the examples below, as well as in our submitted system) most appropriate document chunks to fact-check the provided claim within this knowledge store.

3.1.2 Angle-optimized embedding search

Despite each article in any knowledge store only needing to be compared *once* with its *one specific* claim, which should be the use-case for CrossEncoder reranking (Déjean et al., 2024), our empirical preliminary experiments made us favour a *cosine-similarity* search based on vector embeddings instead. It takes less time to embed the whole knowledge store into vectors than to match each document against a claim using crossencoder, and the produced embeddings can be re-used across experiments.

[3] https://serpapi.com/
[4] https://commoncrawl.org/
[5] https://fever.ai/dataset/averitec.html
[6] The numbers are orientational and were computed on knowledge stores provided for the AVeriTeC dev set.

For our proof of concept, we explore the MTEB (Muennighoff et al., 2023) benchmark leaderboard, looking for a reasonably-sized open-source embedding model, ultimately picking Mixedbread's mxbai-large-v1 (Li and Li, 2024; Lee et al., 2024) optimized for the cosine objective fitting our inteded use.

To reduce querying time at a reasonable exactness tradeoff, we use Faiss index (Douze et al., 2024; Johnson et al., 2019) to store our vectors, allowing us to only precompute semantical representation once, making the retriever respond rapidly in empirical experiments, allowing a very agile prototyping of novel methods to be used.

3.1.3 Chunking with added context

Our initial experiments with the whole AVeriTeC documents for the Document Retrieval step have revealed a significant weakness – while most documents fit within the input size of the embedding model, outliers are common, often with *hundreds of thousands* characters, exceeding the 512 input tokens with little to no coverage of their content.

Upon further examination, these are typically PDF documents of legislature, documentation and communication transcription – highly relevant sources real fact-checker would scroll through to find the relevant part to refer.

This workflow inspires the use of *document chunk retrieval* as used in (Lewis et al., 2020), commonly paired with RAG. We partition each document into sets of its sentences of combined length of N characters at most. To take advantage of the full input size of the vector embedding model we use for semantical search, we arbitrarily set our bound $N = 512 * 4 = 2048$, where 512 is the input dimension of common embedding models, 4 often being used as a rule-of-thumb number of characters per token for US English in modern tokenizers (OpenAI, 2023).

Importantly, each chunk is assigned metadata – the source URL, as well as the full text of the next and previous chunk within the same document. This way, chunks can be presented to the LLM along with their original context in the generation module, where the length constraint is much less of an issue than in vector embedding. As shown in (Drchal et al., 2023), fact-checking models benefit from being exposed to larger pieces of text such as paragraphs or entire documents rather than out-of-context sentences. Splitting our data into the maximum chunks that fit our retrieval model and providing them with additional context may help down the line, preventing the RAG sources from being semantically incomplete.

3.1.4 Pruning the chunks

While the chunking of long articles prevents their information from getting lost to retriever, it makes its search domain too large to embed on demand. As each of the thousands of claims has its own knowledge store, each of possibly tens of thousands of chunks, we seek to omit the chunks having little to no common tokens with our claim using an efficient BM25 (Robertson et al., 1995) search for the nearest ω chunks, setting the ω to 6000 for dev and 2000 for test claims. This yields a reasonably-sized document store for embedding each chunk into a vector, taking an average of 40 s to compute and store using the method described in Section 3.1.2 for each dev-claim using our Tesla V100 GPU.

This allows a quick and agile production of vectorstores for further querying and experimentation, motivated by the AVeriTeC test data being published just several days before the announced submission deadline. The pruning also keeps the resource intensity moderate for real-world applications. However, if time is not of the essence, the step can be omitted.

3.1.5 Diversifying sources: MMR

Our choice of embedding search based on the entire claim rather than generating "search queries" introduces less noise and captures the semantics of the whole claim. It is, however, prone to redundancy among search results, which we address using a reranking by the results' Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998), a metric popular for the RAG task, which maximizes the search results' score computed as (for $D_i \in P$)

$$\lambda \cdot \mathrm{Sim}(D_i, Q) - (1 - \lambda) \cdot \max_{D_j \in S} \mathrm{Sim}(D_i, D_j)$$

Sim denoting the cosine-similarity between embeddings, Q being the search query, and P the prefetched set of documents (by a search which simply maximizes their Sim to Q), forming S as the final search result, by adding each D_i as MMR-argmax one by one, until reaching its desired size.

In our system, we set $\lambda = 0.75$ to favour relevancy rather than diversity, $|S| = 10$ and $|P| = 40$, obtaining a set of diverse sources relevant to each claim at a fraction of cost and complexity of a query-generation driven retrieval, such as that used in (Schlichtkrull et al., 2024).

3.2 Evidence & label generator

The second and the last module on our proposed pipeline for automated fact checking is the Evidence & Label Generator, which receives a claim and k sources (document chunks), and returns l (in our case, $l = 10$) question-answer pairs of evidence abstracted from the sources, along with the veracity verdict – in AVeriTeC dataset, a claim may be classified as *Supported*, *Refuted*, *Not Enough Evidence*, or *Conflicting Evidence/Cherrypicking* with respect to its evidence.

Our approach leverages a Large Language Model (LLM), instructing it to output both evidence and the label in a single step, as a chain of thought. We rely on JSON-structured output generation with source referencing using a numeric identifier, we estimate the label confidences using Likert-scale ratings. The full system prompt can be examined in Listing 1 in Appendix A, and this section further explains the choices behind it.

3.2.1 JSON generation

To be able to collect LLM's results programmatically, we exploit their capability to produce structured outputs, which is on the rise, with datasets available for tuning (Tang et al., 2024) and by the time of writing of this paper (August 2024), systems for strictly structured prediction are beginning to be launched by major providers (OpenAI, 2024).

Despite not having access to such structured-prediction API by the time of AVeriTeC shared task, the current generation of models examined for the task (section 3.2.6) rarely strays from the desired format if properly explained within a system prompt – we instruct our models to output a JSON of pre-defined properties (see prompt Listing 1 in Appendix A) featuring both evidence and the veracity verdict for a given claims.

Although we implement fallbacks, less than 0.5% of our predictions threw a parsing exception throughout experimentation, and could be easily recovered using the same prompting again, exploiting the intrinsic randomness of LLM predictions.

3.2.2 Chain-of-thought prompting

While JSON dictionary should be order-invariant, we can actually exploit the order of outputs in our output structure to make LLMS like GPT-4o output better results (Wei et al., 2024). This is commonly referred to as the "chain-of-thought" prompting – if we instruct the autoregressive LLM to first output the evidence (question, then answer), then a set of all labels with their confidence ratings (see section 3.2.5) and only then the final verdict, its prediction is both cheaper as opposed to implementing an extra module, as well as more reliable, as it must attend to all of the intermediate steps as well.

3.2.3 Source referring

To be able to backtrack the generated evidence to the urls of the used sources, we simply augment each question-answer pair with a source field. We assign a 1-based index[7] to each of the sources to facilitate tokenization and prompt the LLM to refer it as the source ID with each evidence it generates. While hallucination can not be fully prevented, it is less common than it may appear – with RAG gaining popularity, the models are being trained to cite their sources using special citation tokens (Menick et al., 2022), not dissimilarly to our proposal.

3.2.4 Dynamic few-shot learning

To utilise the few-shot learning framework (Brown et al., 2020) shown to increase quality of model output, we provide our LLMs with examples of what we expect the model to do. To obtain such examples, our evidence generator looks up the AVeriTeC train set using BM25 to get the 10 most similar claims, providing them as the few-shot examples, along their gold evidence and veracity verdicts. Experimentally, we also few-shot our models to output an *answer type* (*Extractive*, *Abstractive*, *Boolean*,...) as the *answer type* is listed with each sample anyways, and we have observed its integration into the generation task to slightly boost our model performance.

3.2.5 Likert-scale label confidences

Despite modern LLMs being well capable of predicting the label in a "pick one" fashion, research applications such as ours may prefer them to output a probability distribution over all labels for two reasons.

Firstly, it measures the confidence in each label, pinpointing the edge-cases, secondly, it allows ensembling the LLM classification with any other model, such as Encoders with classification head finetuned on the task of Natural Language Inference (NLI) (see section 4.3).

As the LLMs and other token prediction schemes struggle with the prediction of continuous numbers

[7] We chose the 1-based source indexing to exploit the source-referring data in LLM train set such as Wikipedia, where source numbers start with 1. The improvement in quality over 0-based indexing was not experimentally tested.

which are notoriously hard to tokenize appropriately (Golkar et al., 2023), we come up with a simple alternative: instructing the model to print each of the 4 possible labels, along with their Likert-scale rating: 1 for "strongly disagree", 2 for "disagree", 3 for "neutral", 4 for "agree" and 5 for "strongly agree" (Likert, 1932).

On top of the ease of tokenization, Likert scale's popularity in psychology and other fields such as software testing (Joshi et al., 2015) adds another benefit – both the scale itself and its appropriate usage were likely demonstrated many times to LLMs during their unsupervised training phase.

To convert the ratings such as `{"Supported":2, "Refuted":5, "Cherrypicking":4, "NEE":2}` to a probability distribution, we simply use softmax (Bridle, 1989). While the label probabilities are only emulated (and may only take a limited, discrete set of values) and the system may produce ties, it gets the job done until further research is carried out.

3.2.6 Choosing LLM

In our experiments, we have tested the full set of techniques introduced in this section, computing the text completion requests with:

1. GPT-4o (version `2024-05-13`)
2. Claude-3.5-Sonnet (`2024-06-20`), using the Google's Vertex API
3. LLaMA 3.1 70B, in the final experimets to see if the pipeline can be re-produced using open-source models

Their comparison can be seen in tables 1 and 2; for our submission in the AVeriTeC shared task, GPT-4o was used.

4 Other examined approaches

In this section, we also describe a third, optional module we call the *veracity classifier*, which takes the claim and its evidence generated by our evidence & label generator (section 3.2) and predicts the veracity label independently, based on the suggested evidence, using a fine-tuned NLI model. We also describe the options of its ensembling with veracity labels predicted in the generative step (section 3.2.5).

The absence of a dedicated veracity classifier has not been shown to decrease the performance of our pipeline significantly (as shown, e.g., in tables 2 and 1) so we suggest to omit this step altogether and we proceed to participate in the AVeriTeC shared task without it, proposing a clean and simple RAG pipeline without the extra step (Figure 1) for the fact-checking task.

4.1 Single-evidence classification with label aggregation

In the earliest stages of experimenting, we utilized the baseline classifier provided by AVeriTeC authors[8] (Schlichtkrull et al., 2024). It is based on the BERT (Devlin et al., 2019) and was further fine-tuned on the AVeriTeC dataset (Schlichtkrull et al., 2024). It takes one claim and one question-answer evidence as input – each claim therefore has multiple classifications, one for each evidence. The classifications are then aggregated using a heuristic of several if-clauses to determine the final label.

We experiment with altering this heuristic (e.g. by making *not enough evidence* the final label only when no other labels are present at any evidence), and training NLI models that could work better with it, such as 3-way DeBERTaV3 (He et al., 2023) without a breakthrough result, motivating a radically different approach.

4.2 Multi-evidence classification

The multi-evidence approach is to fine-tune a 4-way Natural Language Inference (NLI) classifier, using the full scope of evidence directly at once, without heuristics. For that, we concatenate all of the evidence together using a separator `[SEP]` token. This allows the model to know exact question-answer borders, albeit using a space has turned out to be just as accurate as the experiments went on. As the veracity verdict should be independent of the evidence ordering, we also experiment with sampling different permutations in the fine-tuning step to increase the size of our data.

We carry out the fine-tuning using the AVeriTeC train split with gold evidence and labels on DeBERTaV3 (He et al., 2023) in two variants: the original large one[9] and one pre-finetuned on NLI tasks[10], and also Mistral-7B-v0.3 model[11] with a classification head (MistralForSequenceClassification) provided by the Huggingface Transformers

[8] https://huggingface.co/chenxwh/AVeriTeC

[9] https://huggingface.co/microsoft/deberta-v3-large

[10] https://huggingface.co/cross-encoder/nli-deberta-v3-large

[11] https://huggingface.co/mistralai/Mistral-7B-v0.3

library (Wolf et al., 2020) that utilizes the last token. In the preliminary testing phase, the original DeBERTaV3 Large performed the best and was used in all other experimental settings.

From the approaches described above, we achieved the best results for the development split with gold evidence and labels with a model without permuting the evidence, achieving 0.71 macro F_1 score using a space-separation. The `[SEP]` model achieved a comparable 0.70 macro F_1 score, and the random order model performed worse with a 0.67 macro F_1 score, all improving significantly upon baseline, yet falling behind the capabilities of generating the labels alongside evidence in a single chain-of-thought. We provide our best DeBERTaV3 finetuned model publicly in a Huggingface repository[12].

4.3 Ensembling classifiers

Encouraged by the promising results of our multi-evidence classifiers, we go on to try to ensemble the models with LLM predictions from section 3.2.5, using a weighted average of the class probabilities of our models. We have experimented with multiple weight settings: 0.5:0.5 for even votes, 0.3:0.7 in favour of the LLM to exploit its accuracy while tipping its scales in cases of a more spread-out label probability distribution, as well as 0.1:0.9 to use the fine-tuned classifier only for tie-breaking, listing the results in Table 1.

We also tried tuning our ensemble weights based on a subset of the dev split, without a breakthrough in accuracy on the rest of dev samples.

The last method we tried was stacking using logistic regression. However, this setup classified no labels from *Not Enough Evidence* and *Conflicting Evidence/Cherrypicking*, and we could not achieve reasonable results. For logistic regression, we used the scikit-learn library (Pedregosa et al., 2011).

We conclude that the augmentation of the pipeline from Figure 1 with a classification module using a single NLI model or an ensemble with LLM is unneccessary, as it adds complexity and computational cost without paying off on the full pipeline performance (Table 2).

4.4 Conflicting Evidence/Cherrypicking detection

During the experiments, we discovered that classifying the *Conflicting Evidence/Cherrypicking* class is the most challenging task, achieving a near-zero F_1-score across our various prototype pipelines. To overcome this problem, we tried to build a binary classifier with cherrypicking as positive class. We tried to use the DeBERTaV3 Large model with both basic and weighted cross-entropy loss (other experimental settings were the same as in section 4.2), but it could not pick up the training task due to the *Conflicting Evidence/Cherrypicking* underrepresentation in train set – less than 7% of the samples carry the label.

Even after exploring various other methods, we did not get a reliable detection scheme for this task, perhaps motivating a future collection of data that represents the class better. While writing this system description paper, we found an interesting research by Jaradat et al. (2024) that uses a radically different approach to detect cherrypicking in newspaper articles.

5 Results and analysis

We examine our pipeline results using two sets of metrics – firstly, we measure the prediction accuracy and F_1 over predict labels without any ablation, that is obtaining predicted labels using the predicted evidence generated on top the predicted retrieval results. While the retrieval module is fixed throughout the experiment (a full scheme described in section 3.1), various Evidence & Label generators and classifiers are compared in Table 1, showcasing their performance on the same sources. The results show that if we disregard the quality of evidence, models are more or less interchangeable, without a clear winner across the board – an ensemble of DeBERTA and Claude-3.5-Sonnet gives the best F_1 score, while GPT-4o scores 72% accuracy.

In real world, however, the evidence quality is critical for the fact-checking task. We therefore proceed to estimate it using the hu-METEOR evidence question score, QA score and AVeriTeC score benchmarks briefly explained in Section 2 and in greater detail in (Schlichtkrull et al., 2024). We use the provided AVeriTeC scoring script to calculate the values for Table 2, using its EvalAI blackbox to obtain the test scores without seeing the gold test data.

The latter experiments shown in Table 2 suggests the superiority of GPT-4o to predict the results for our pipeline with a margin. Even if we simplify the evidence & label generation step by omitting the

[12] https://huggingface.co/ctu-aic/deberta-v3-large-AVeriTeC-nli

Classifier	Acc	F_1	Prec.	Recall
GPT4o	0.72	0.46	0.48	0.47
Claude 3.5 Sonnet	0.64	0.49	0.50	0.52
DeBERTa	0.63	0.39	0.40	0.41
DeBERTa - random@10	0.65	0.41	0.41	0.44
$0.5 \cdot$ DeBERTa $+ 0.5 \cdot$ GPT4o	0.70	0.43	0.41	0.45
$0.5 \cdot$ DeBERTa $+ 0.5 \cdot$ Claude	0.68	0.47	0.50	0.49
$0.3 \cdot$ DeBERTa $+ 0.7 \cdot$ GPT4o	0.72	0.45	0.45	0.46
$0.3 \cdot$ DeBERTa $+ 0.7 \cdot$ Claude	0.66	**0.50**	**0.51**	**0.53**
$0.1 \cdot$ DeBERTa $+ 0.9 \cdot$ GPT4o	0.72	0.39	0.46	0.43
$0.1 \cdot$ DeBERTa $+ 0.9 \cdot$ Claude	0.64	0.49	0.50	0.54
Llama 3.1	**0.73**	0.44	0.43	0.46

Table 1: Evalution of the label generators, classifier models and their ensembles on the AVeriTeCdevelopment set. F_1, Precision and Recall are computed as macro-averages. The random@10 suffix indicates that the classifier used average of 10 different random orders of QA pairs for each claim. GPT4o stands for the Likert classifier based on GPT-4o, Claude 3.5 Sonnet is the Likert classifier based on Claude 3.5 Sonnet, and DeBERTa is classifier based on DeBERTaV3 Large fine-tuned on AVeriTeC gold evidence and labels.

dynamic few-shot learning (section 3.2), answer-type tuning and Likert-scale confidence emulation, it still scores above others, also showing that our pipeline can be further simplified when needed. Regardless of the LLM in use, the results of our pipeline improve upon the AVeriTeC baseline dramatically.

Posterior to the original experiments and to the AVeriTeC submission deadline, we also compute the pipeline results using an open-source model – the Llama 3.1 70B[13] (Dubey et al., 2024) obtaining encouraging scores, signifying our pipeline being adaptable to work well without the need to use a blackboxed proprietary LLM.

5.1 API costs

During our experimentation July 2024, we have made around 9000 requests to OpenAI's `gpt-4o-2024-05-13` batch API, at a total cost of $363. This gives a mean cost estimate of $0.04 per a single fact-check (or $0.08 using the API without the batch discount) that can be further reduced using cheaper models, such as `gpt-4o-2024-08-06`.

We argue that such costs make our model suitable for further experiments alongside human fact-checkers, whose time spent reading through each source and proposing each evidence by themselves would certainly come at a higher price.

Our successive experiments with Llama 3.1 (Dubey et al., 2024) show promising results as well, nearly achieving parity with GPT. The use of open-source models such as LLaMa or Mistral allows running our pipeline on premise, without leaking data to a third party and billing anything else than the computational resources. For further experiments, we are looking to integrate them into the attached Python library using VLLM (Kwon et al., 2023).

5.2 Error analysis

In this section, we provide the results of an explorative analysis of 20 randomly selected samples from the development set. We divide our description of the analysis into the pipeline and dataset errors.

5.2.1 Pipeline errors

Our pipeline tends to rely on unofficial (often newspaper) sources rather than official government sources, e.g., with a domain ending or containing `gov`. On the other hand, it seems that the annotators prefer those sources. This could be remedied by implementing a different source selection strategy, preferring those official sources. For an example, see Listing 2 in Appendix B.

Another thing that could be recognised as an error is that our pipeline usually generates all ten allowed questions (upper bound given by the task (Schlichtkrull et al., 2024)). The analysis of the samples shows that the last questions are often unrelated or redundant to the claim and do not contribute directly to better veracity evaluation. However, since the classification step of our pipeline is not dependent on the number of question-answer pairs, this is not a critical error. Listing 3 in Appendix B shows an example of a data point with some unrelated questions.

When the pipeline generates extractive answers, it sometimes happens that the answer is not precisely extracted from the source text but slightly modified. An example of this error can be seen in Listing 4 in Appendix B. This error is not critical, but it could be improved in future works, e.g. using post-processing via string matching.

Individual errors were also caused by the fact that we do not use the claim date in our pipeline and because our pipeline cannot analyse PDFs with tables properly. The last erroneous behaviour we have noticed is that the majority of questions and

[13] https://huggingface.co/hugging-quants/Meta-Llama-3.1-70B-Instruct-AWQ-INT4

Pipeline Name	Dev Set Scores			Test Set Scores		
	Q only	Q+A	AVeriTeC	Q only	Q+A	AVeriTeC
GPT-4o (full-featured pipeline)	**0.46**	**0.29**	**0.42**	0.46	**0.32**	**0.50**
GPT-4o (simplified pipeline)	0.45	0.28	0.38	0.45	0.30	0.47
Claude-3.5-Sonnet (full-featured)	0.43	0.28	0.35	0.42	0.30	0.46
GPT-4o (with DeBERTa classification)	0.45	0.28	0.36	–	–	–
AVeriTeC baseline	0.24	0.19	0.09	0.24	0.20	0.11
Llama 3.1 70B (full-featured)	**0.46**	0.27	0.36	**0.47**	0.29	0.42

Table 2: Comparison of Pipeline Scores on Dev and Test Sets. Q, Q+A are Hu-METEOR scores against gold data, AVeriTeC scores are calculated as referred in section 2 thresholded at 0.25. "Full-featured" pipelines use the all the improvement techniques introduced in section 3, while the simplified pipeline omits the dynamic few-shot learning, answer-type-tuning and Likert-scale confidence emulation described in section 3.2

answers are often generated from a single source. This should not be viewed as an error, but by introducing diversity into the sources, the pipeline would be more reliable when deployed in real-world scenarios.

5.2.2 Dataset errors

During the error analysis of our pipeline, we also found some errors in the AVeriTeC dataset that we would like to mention. In some cases, there is a leakage of PolitiFact or Factcheck.org fact-checking articles where the claim is already fact-checked. This leads to a situation where our pipeline gives a correct verdict using the leaked evidence. However, annotators gave a different label (often Not Enough Evidence). An example of this error is shown in Listing 5 in Appendix B.

Another issue we have noticed is the inconsistency in the questions and answers given by annotators. Sometimes, they tend to be longer, including non-relevant information, while some are much shorter, as seen in Listing 6 in Appendix B. The questions are often too general, or the annotators seem to use outside knowledge. This inconsistency in the dataset leads to a decreased performance of any models evaluated on this dataset.

5.2.3 Summary

Despite the abovementioned errors, the explorative analysis revealed that our pipeline consistently gives reasonable questions and answers for the claims. Most misclassified samples in those 20 data points were due to dataset errors.

6 Conclusion

In this paper, we describe the use and development of a RAG pipeline over real world claims and data scraped from the web for the AVeriTeC shared task. Its main advantage are its simplicity, consisting of just two decoupled modules – Retriever and an Evidence & Label Generator – and leveraging the trainable parameters of a LLM rather than on complex pipeline engineering. The LLMs capabilities may further improve in future, making the upgrades of our system trivial.

In section 3, we describe the process of adding features to both modules well in an iterative fashion, describing real problems we have encountered and the justifications of their solution, hoping to share our experience on how to make such systems robust and well-performing. We publish our failed approaches in section 4 and the metrics we observed to benchmark our systems in section 5. We release our Python codebase to facilitate further research and applications of our system, either as a baseline for future research, or for experimenting alongside human fact-checkers.

6.1 Future works

1. Integrating a search API for use in real-world applications

2. Re-examine the Likert-scale rating (section 3.2.5) to establish a more appropriate and fine-grained means of tokenizing the label probabilities

3. Generating evidence in the form of declarative sentences rather than Question-Answer pairs should be explored to see if it leads for better or worse fact-checking performance

4. RAG-tuned LLMs such as those introduced in (Menick et al., 2022) could be explored to see if they offer a more reliable source citing

Limitations

The evaluation of our fact-checking pipeline is limited to the English language and the AVeriTeC dataset (Schlichtkrull et al., 2024). This is a severe limitation as the pipeline when deployed in a real-world application, would encounter other languages and forms of claims not covered by the used dataset.

Another limitation is that we are using a large language model. Because of that, future usage is limited to using an API of a provider of LLMs or having access to a large amount of computational resources, which comes at significant costs. Using APIs also brings the disadvantage of sending data to a third party, which might be a security risk in some critical applications. LLM usage also has an undeniable environmental impact because of the vast amount of electricity and resources used.

The reliability of the generated text is a limitation that is often linked to LLMs. LLMs sometimes hallucinate (in our case, it would mean using sources other than those given in the system prompt), and they can be biased based on their extensive training data. Moreover, because of the dataset size, it is impossible to validate each output of the LLM, and thus, we are not able to 100% guarantee the quality of the results.

Ethics statement

It is essential to note that our pipeline is not a real fact-checker that could do a human job but rather a study of future possibilities in automatic fact-checking and a showcase of the current capabilities of state-of-the-art language models. The pipeline in its current state should only be used with human supervision because of the potential biases and errors that could harm the consumers of the output information or persons mentioned in the claims. The pipeline could be misused to spread misinformation by directly using misinformation sources or by intentionally modifying the pipeline in a way that will generate wrong outputs.

Another important statement is that our pipeline was in its current form explicitly built for the AVeriTeC shared task, and thus, the evaluation results reflect the bias of the annotators. For more information, see the relevant section of the original paper (Schlichtkrull et al., 2024).

The carbon costs of the training and running of our pipeline are considerable and should be taken into account given the urgency of climate change. At the time of deployment, the pipeline should be run on the smallest possible model that can still provide reliable results, and the latest hardware and software optimisations should be used to minimise the carbon footprint.

Acknowledgements

We would like to thank Bryce Aaron from UNC for exploring the problems of search query generation and pinpointing claims of underrepresented labels using numerical methods that did not make it into our final pipeline but gave us a frame for comparison.

This research was co-financed with state support from the Technology Agency of the Czech Republic and the Ministry of Industry and Trade of the Czech Republic under the TREND Programme, project FW10010200. The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics" is also gratefully acknowledged. We would like to thank to OpenAI for providing free credit for their paid API via Researcher Access Program[14].

References

John Bridle. 1989. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA. Association for Computing Machinery.

[14] https://openai.com/form/researcher-access-program/

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Jan Drchal, Herbert Ullrich, Tomáš Mlynář, and Václav Moravec. 2023. Pipeline and dataset generation for automated fact-checking in almost any language.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, ..., and Zhiwei Zhao. 2024. The llama 3 herd of models.

Hervé Déjean, Stéphane Clinchant, and Thibault Formal. 2024. A thorough comparison of cross-encoders and llms for reranking splade.

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, Bruno Régaldo-Saint Blancard, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. 2023. xval: A continuous number encoding for large language models.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing.

Israa Jaradat, Haiqi Zhang, and Chengkai Li. 2024. On context-aware detection of cherry-picking in news reporting.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Ankur Joshi, Saket Kale, Satish Chandel, and Dinesh Pal. 2015. Likert scale: Explored and explained. *British Journal of Applied Science & Technology*, 7:396–403.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Sean Lee, Aamir Shakir, Darius Koenig, and Julius Lipp. 2024. Open source strikes bread - new fluffy embeddings model.

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 *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Xianming Li and Jing Li. 2024. AoE: Angle-optimized embeddings for semantic textual similarity. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1825–1839, Bangkok, Thailand. Association for Computational Linguistics.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):55.

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, and Nat McAleese. 2022. Teaching language models to support answers with verified quotes.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

OpenAI. 2023. What are tokens and how to count them? Accessed: 15 August 2024.

OpenAI. 2024. Introducing structured outputs in the api. Accessed: 15 August 2024.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1995. Okapi at trec-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*.

Michael Schlichtkrull, Zhijiang Guo, and Andreas Vlachos. 2024. Averitec: a dataset for real-world claim verification with evidence from the web. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Xiangru Tang, Yiming Zong, Jason Phang, Yilun Zhao, Wangchunshu Zhou, Arman Cohan, and Mark Gerstein. 2024. Struc-bench: Are large language models really good at generating complex structured data?

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018a. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018b. 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.

Herbert Ullrich, Jan Drchal, Martin Rýpar, Hana Vincourová, and Václav Moravec. 2023. Csfever and ctkfacts: acquiring czech data for fact verification. *Language Resources and Evaluation*, 57(4):1571–1605.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 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, Online. Association for Computational Linguistics.

A System prompt

```
You are a professional fact checker, formulate up to 10 questions that cover all
  the facts needed to validate whether the factual statement (in User message) is
  true, false, uncertain or a matter of opinion. Each question has one of four
  answer types: Boolean, Extractive, Abstractive and Unanswerable using the
  provided sources.
After formulating Your questions and their answers using the provided sources, You
  evaluate the possible veracity verdicts (Supported claim, Refuted claim, Not
  enough evidence, or Conflicting evidence/Cherrypicking) given your claim and
  evidence on a Likert scale (1 - Strongly disagree, 2 - Disagree, 3 - Neutral, 4 -
  Agree, 5 - Strongly agree). Ultimately, you note the single likeliest veracity
  verdict according to your best knowledge.
The facts must be coming from these sources, please refer them using assigned IDs:
---
## Source ID: 1 [url]
[context before]
[page content]
[context after]
...

---
## Output formatting
Please, you MUST only print the output in the following output format:
```json
{
 "questions":
     [
         {"question": "<Your first question>", "answer": "<The answer to the Your
           first question>", "source": "<Single numeric source ID backing the
           answer for Your first question>", "answer_type":"<The type of first
           answer>"},
         {"question": "<Your second question>", "answer": "<The answer to the Your
           second question>", "source": "<Single numeric Source ID backing the
           answer for Your second question>", "answer_type":"<The type of second
           answer>"}
     ],
 "claim_veracity": {
     "Supported": "<Likert-scale rating of how much You agree with the 'Supported'
       veracity classification>",
     "Refuted": "<Likert-scale rating of how much You agree with the 'Refuted'
       veracity classification>",
     "Not Enough Evidence": "<Likert-scale rating of how much You agree with the
       'Not Enough Evidence' veracity classification>",
     "Conflicting Evidence/Cherrypicking": "<Likert-scale rating of how much You
       agree with the 'Conflicting Evidence/Cherrypicking' veracity classification>"
 },
 "veracity_verdict": "<The suggested veracity classification for the claim>"
}
```

Few-shot learning
You have access to the following few-shot learning examples for questions and
  answers.:

Question examples for claim "{example["claim"]}" (verdict
  {example["gold_label"]})
"question": "{question}", "answer": "{answer}", "answer_type": "{answer_type}"
...
```

Listing 1: System prompt for the LLMs, AVeriTeC claim is to be entered into the user prompt. Three dots represent omitted repeating parts of the prompt.

B Examples of errors

```
Claim 479: Donald Trump said "When the
  anarchists started ripping down our statues
  and monuments, right outside, I signed an
  order immediately, 10 years in prison."

gold evidence example:
    question: What was the law signed by Trump
      regarding damaging federal property?
    answer: Trump signed an executive order that
      authorizes a penalty of up to 10 years in
      prison for damaging federal property. It
      does not say that it will automatically be
      a 10 year prison sentence.,
    source_url: https://web.archive.org/web
      /20210224033536/https://www.whitehouse.gov/
      presidential-actions/executive-order-
      protecting-american-monuments-memorials-
      statues-combating-recent-criminal-violence/

pipeline evidence example:
    question: Did Trump sign an order related to
      vandalism of statues and monuments?,
    answer: Yes, Trump signed an executive order
      to prosecute those who damage national
      monuments, making it a punishable offense
      with up to 10 years in jail.,
    url: https://m.economictimes.com/news/
      international/world-news/trump-makes-
      vandalising-national-monuments-punishable-
      offence-with-up-to-10-yrs-jail/articleshow
      /76658610.cms
```

Listing 2: Example of a claim where our pipeline uses newspaper sources instead of official government sources.

```
Claim 295: Trump campaign asked Joe Biden to
  release a list of potential Supreme Court
  picks only after Ginsburg's passing
question 1: Did Joe Biden claim that the Trump
  campaign asked him to release a list of
  potential Supreme Court picks only after
  Ginsburg's passing?
question 2: Did the Trump campaign ask Joe Biden
   to release a list of potential Supreme Court
  picks before Ginsburg's passing?
question 3: When did Trump release his latest
  list of potential Supreme Court nominees?
question 4: Did Trump personally demand that
  Biden release a list of potential Supreme
  Court nominees before Ginsburg's death?
question 5: What did Trump say about Biden
  releasing a list of potential Supreme Court
  nominees during the Republican National
  Convention?
question 6: Did the Trump campaign issue a
  statement on September 17, 2020, regarding
  Biden releasing a list of potential Supreme
  Court nominees?
question 7: What did the Trump campaign's
  statement on September 9, 2020, say about
  Biden releasing a list of potential Supreme
  Court nominees?
question 8: Did Biden indicate in June 2020 that
   he might release a list of potential Supreme
  Court picks?
quetion 9: What reason did Biden give for not
  releasing a list of potential Supreme Court
  nominees?,
question 10: Did Biden pledge to nominate a
  Black woman to the Supreme Court?
```

Listing 3: Example of a claim and questions showing that the last tends to be unrelated or redundant to fact-checking of the claim.

```
Claim #155 - Trump said 'there were fine people
  on both side' in far-right protests.
answer: "You had some very bad people in that
  group, but you also had people that were very
  fine people, on both sides.",
answer_type: Extractive
url: https://www.theatlantic.com/politics/
  archive/2017/08/trump-defends-white-
  nationalist-protesters-some-very-fine-people-
  on-both-sides/537012/
scraped text: ... "You also had some very fine
  people on both sides," he said. The Unite the
  Right rally that sparked the violence in
  Charlottesville featured several leading names
   in the white-nationalist alt-right movement,
  and also attracted people displaying Nazi
  symbols. ...
```

Listing 4: Example of a claim where our pipeline did not exactly extract the answer.

```
Claim #483 - Donald Trump said "We have spent
  nearly $2.5 trillion on completely rebuilding
  our military, which was very badly depleted
  when I took office."
Gold Label: Not Enough Evidence
Predicted Label: Refuted
pipeline evidence example:
    question: What is the total defense budget
      for the last four fiscal years under Trump?
    url: https://www.politifact.com/factchecks
      /2020/jan/10/donald-trump/trump-exaggerates
      -spending-us-military-rebuild/
    question: Did Trump spend $2.5 trillion
      specifically on rebuilding the military?
    url: https://www.factcheck.org/2020/07/trumps
      -false-military-equipment-claim/
    ...
```

Listing 5: An example of a claim where the evidence consists mainly of evidence from PolitiFact and Factcheck.org fact-checking articles leading to different predicted label than in the gold dataset

```
Claim #0 - In a letter to Steve Jobs, Sean
  Connery refused to appear in an apple
  commercial.
Gold Evidence:
    question: Where was the claim first published
    answer: It was first published on Sccopertino
    question: What kind of website is Scoopertino
    answer: Scoopertino is an imaginary news
      organization devoted to ferreting out the
      most relevant stories in the world of Apple
      , whether or not they actually occurred -
      says their about page

Claim #315 - The fastest Supreme Court justice
  ever confirmed in the U.S. was 47 days.
Gold Evidence:
    question: What is the quickest time a Supreme
       Court justice nomination has been
      confirmed in the United States?
    answer: John Paul Stevens waited the fewest
      number of days (19)-followed by the most
      recent nominee to the Court, Amy Coney
      Barrett (27).61
    question: What is the average number of days
      between a nomination for a Supreme Court
      justice and the final Senate vote?
    answer: Overall, the average number of days
      from nomination to final Senate vote is
      68.2 days (or approximately 2.2 months),
      while the median is 69.0 days.62 Of the 9
      Justices currently serving on the Court,
      the average number of days from nomination
      to final Senate vote is 72.1 days (or
      approximately 2.4 months), while the median
       is 73.0 days. Among the current Justices,
      Amy Coney Barrett waited the fewest number
      of days from nomination to confirmation
      (27), while Clarence Thomas waited the
      greatest number of days (99).
```

Listing 6: An example of a claims which differs in length.