Memory-Efficient Multi-Stage Fact-Checking System with Dual Encoder Retrieval, In-Context Learning, and Ensemble Selection

Muhammad Md Nasrein and Ni Bin Xin and Dong Xiang
University of Melbourne
Parkville

Abstract

Automated fact-checking requires accurately retrieving and verifying evidence from large corpora to support or refute claims. While dense retrieval and cross-encoder reranking have become standard, these approaches often struggle with memory inefficiency, context fragmentation, and limited reasoning capability in downstream classification.

In this project, we propose a memory-efficient multi-stage fact-checking system combining dual encoder retrieval, in-context learning, and ensemble selection. In **Stage 1**, our system employs a dual encoder retrieval pipeline with dynamic VRAM-safe streaming, enabling large-scale retrieval without reliance on heavy indexing structures such as FAISS, and making it suitable for deployment on memory-constrained devices, followed by cross-encoder reranking for fine-grained evidence selection. In **Stage 2**, we explore few-shot in-context learning using open-source large language models with multiple prompting strategies, including question-answering (Q&A) and chain-of-thought (CoT) reasoning, leveraging the top-k retrieved evidence. Finally, in **Stage 3**, we introduce an ensemble classification approach that selects the best prediction by optimizing the harmonic mean of evidence F-score and claim classification accuracy.

1 Introduction

Fact-checking is a critical task in natural language processing (NLP) to combat misinformation by retrieving and verifying evidence from large corpora (Thorne et al., 2018). While dense retrieval and cross-encoder reranking methods have advanced the field (Karpukhin et al., 2020; Nogueira and Cho, 2019), existing approaches face challenges in scalability, context fragmentation, and limited reasoning capabilities.

Most prior works focus either on retrieval efficiency or claim verification, often neglecting the integration of robust reasoning and balanced performance across both tasks. Additionally, evidence passage chunking can lead to context loss, and end-to-end models struggle to scale in memory-constrained environments.

To address these gaps, we propose a memory-efficient, multi-stage fact-checking system combining: (1) dual encoder retrieval with dynamic streaming; (2) in-context learning using large language models (LLMs) with question-answering (Q&A) and chain-of-thought (CoT) prompting; and (3) ensemble selection based on the harmonic mean of evidence F-score and claim classification accuracy.

Our system is particularly suited for deployment in resource-constrained settings, such as mobile devices, newsrooms with limited compute infrastructure, or civic tech platforms aiming to provide real-time fact-checking at scale without reliance on heavy retrieval indices like Facebook AI Similarity Score (FAISS). Our streaming-based retrieval approach eliminates the need for persistent vector indices and reduces VRAM usage by processing evidence batches on-the-fly, enabling operation under low-memory constraints.

2 Approach

2.1 System Overview

We propose a three-stage fact-checking system designed to address scalability, context preservation, and reasoning integration in claim verification. Figure 1 illustrates the overall architecture.

Stage 1 employs a dual encoder to retrieve evidence passages using VRAM-efficient streaming **dot product similarity**. Retrieved passages are reranked using a cross-encoder, providing top-k evidence for each claim.

Stage 2 applies few-shot in-context learning using a large language model with diverse prompting strategies, including question-answering (Q&A) and chain-of-thought (CoT) reasoning (Wei et al.,

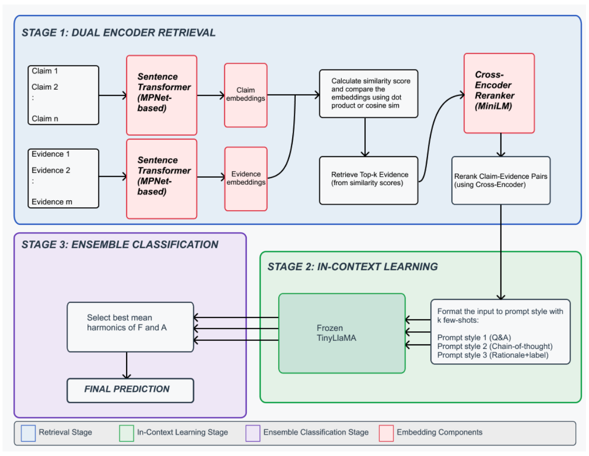
STAGE 1: DUAL ENCODER RETRIEVAL
Claim 1
Claim 2
:
Claim n
Sentence Transformer (MPNet-based)
Claim embeddings
Evidence 1
Evidence 2
:
Evidence m
Sentence Transformer (MPNet-based)
Evidence embeddings
Calculate similarity score and compare the embeddings using dot product or cosine sim
Retrieve Top-k Evidence (from similarity scores)
Cross-Encoder Reranker (MiniLM)
Rerank Claim-Evidence Pairs (using Cross-Encoder)
STAGE 3: ENSEMBLE CLASSIFICATION
Select best mean harmonics of F and A
FINAL PREDICTION
STAGE 2: IN-CONTEXT LEARNING
Frozen TinyLlaMA
Format the input to prompt style with k few-shots:
Prompt style 1 (Q&A)
Prompt style 2 (Chain-of-thought)
Prompt style 3 (Rationale+label)
Retrieval Stage
In-Context Learning Stage
Ensemble Classification Stage
Embedding Components

Figure 1: Overview of the proposed multi-stage fact-checking system.

2022), enabling reasoning-aware claim verification.

Stage 3 performs ensemble selection by optimizing the harmonic mean of evidence F-score and claim classification accuracy using outputs from Stage 2. This ensures balanced performance across retrieval and verification.

The modular design supports independent evaluation of each stage while enabling integration into a unified, memory-efficient fact-checking pipeline suitable for resource-constrained deployments.

2.2 Stage 1: Dual Encoder Retrieval

Stage 1 implements a dual encoder retrieval pipeline where claims and evidence passages are independently encoded into dense vector representations using the `multi-qa-mpnet-base-dot-v1` model.

Retrieval is performed using a VRAM-efficient streaming approach, where evidence embeddings are processed in buffered batches and **dot product similarity** is computed directly on the GPU. This avoids the need for large vector indices and supports retrieval at scale under constrained hardware settings. This method reduces peak memory load and avoids full index storage in RAM, allowing retrieval to run within approximately 4GB of VRAM with 32-64 batches on corpora exceeding 100k passages. This design choice also simplifies the deployment on low-end GPUs or devices without disk-based retrieval frameworks.

The top-k retrieved evidence passages are reranked using a cross-encoder (`cross-encoder/ms-marco-MiniLM-L-12-v2`) that jointly encodes claim-evidence pairs and produces refined relevance scores. This reranking step enhances precision by considering cross-attentive interactions between the claim and each candidate evidence.

2.3 Stage 2: In-Context Learning with Multi-Prompt Reasoning

Stage 2 performs claim verification using few-shot in-context learning with a large language model (LLM). The system leverages the top-k retrieved evidence passages from Stage 1 as input to the LLM, applying diverse prompting strategies to encourage reasoning diversity and reduce bias from single prompt formulations.

Two prompting styles are employed: (1) question-answering (Q&A) prompting, which formulates the claim and evidence as a direct question for the model to answer; and (2) chain-of-thought (CoT) prompting (Wei et al., 2022), which guides the model to reason step-by-step before providing a final label. Multiple prompts are generated for each claim, producing diverse predictions from different reasoning paths.

2.4 Stage 3: Ensemble Selection Based on Harmonic Mean

Stage 3 aggregates the predictions generated from Stage 2 using an ensemble selection mechanism. For each claim, the system evaluates candidate predictions by computing the harmonic mean of evidence retrieval F-score and claim classification accuracy, as measured using the provided `eval.py` script.

The prediction achieving the highest harmonic mean is selected as the final system output. This strategy ensures that the system balances both evidence relevance and label correctness, mitigating the trade-off between retrieval recall and classification precision.

3 Baseline

3.1 Baseline

As a baseline, we implement a standard dual encoder retrieval system using the `all-MiniLM-L6-v2` model from Sentence-Transformers. In this setup, claims and evidence passages are independently encoded into dense vector representations. Retrieval is performed by computing **cosine similarity** between the claim and evidence embeddings.

The baseline system processes the entire claim and evidence texts without any preprocessing. All embeddings are computed in batches and stored for efficient similarity computation using `sklearn`'s `cosine_similarity` function. The system retrieves the top-k evidence passages for each claim based on the similarity scores.

This baseline does not incorporate reranking, reasoning mechanisms, or ensemble strategies, providing a straightforward retrieval-only pipeline for comparison against our proposed multi-stage system.

4 Experiments

4.1 Evaluation Method

To evaluate the effectiveness of different evidence selection and claim classification strategies using large language models (LLMs), we adopt three standard metrics to evaluate our model performance: (1) `Evidence Retrieval F1 score (F)`, measuring how well the retrieved evidence matches the ground truth; (2) `Claim Classification Accuracy (A)`, assessing how accurately the model classifies the claim; and (3) `The harmonic mean of F and A`, providing a balanced evaluation of both retrieval and classification components.

4.2 Experimental Setup

Dataset. We use three datasets: `train_claims.json` and `dev_claims.json` contain labeled claims with corresponding labels in `SUPPORTS`, `REFUTES`, `NOT_ENOUGH_INFO`, `DISPUTED`; `evidence.json` contains a large collection of textual evidence passages serving as the knowledge source. The task is to build an automated fact-checking system that, given a claim, retrieves relevant evidence from `evidence.json` and classifies the claim into one of the four categories based on the retrieved content.

The experiment is divided into three main stages:

Stage 1: Evidence Retrieval. Given an input `claim_text`, the goal is to retrieve the top-k relevant evidence passages from `evidence.json`. We use cosine similarity as our baseline retrieval method and explore the following configurations:

- **Preprocessing:** Whether to pre-process the claim text and evidence text (e.g., stopword removal, lowercase, lemmatize).
- **Reranking:** Whether to apply a reranking module (using cross-encoder model to reorder the top-k results).

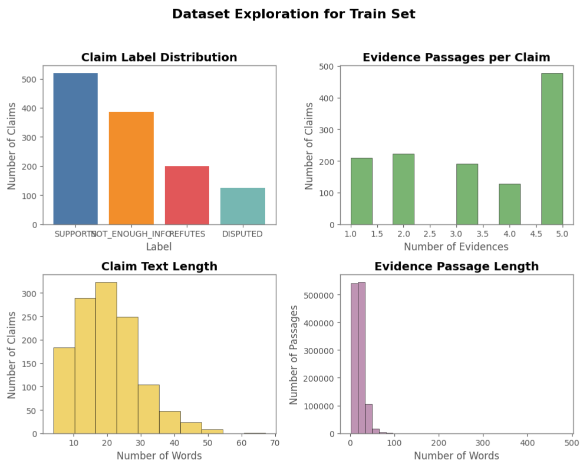

Figure 2: Data Distribution of Training set.

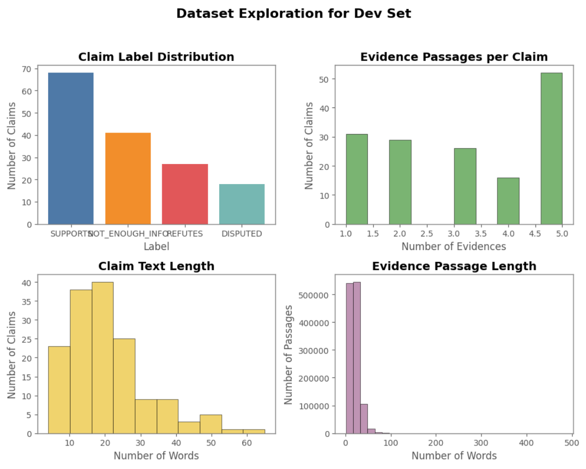

Figure 3: Data Distribution of Dev set.

- **Fine-tuning:** Whether to fine-tune the retriever on the task-specific dataset.
- **Kt (retrieval):** The number of evidence passages selected in the top-k.
- **Kr (reranking):** The number of candidates reranked to choose the final evidence set.

Stage 2: Claim Classification. Stage 2 is built upon the most optimal results from Stage 1 (**Kt**=,**Kr**=3). We use the `TinyLlama/TinyLlama-1.1B-Chat-v1.0` model under a few-shot in-context learning setup. Each input consists of a `claim_text` and its retrieved evidence passages. The model outputs a label from the set: `SUPPORTS`, `REFUTES`, `DISPUTED`, `NOT_ENOUGH_INFO`. We evaluate the following three model variants:

- **Model 1:** For each test claim, retrieves the most similar claims from train_claims.json using cosine similarity. Their associated evidence and labels are used as few-shot examples in the prompt.

- **Model 2:** Extends Model 2 with Chain-of-Thought (CoT) prompting to encourage intermediate reasoning steps before generating the final label.
- **Model 3:** Uses a fixed set of examples from train_claims.json as few-shot exemplars without dynamic selection or additional strategies.

We experiment with different numbers of few-shot examples (from 1 to 5). All models are evaluated on both train_claims.json and dev_claims.json. Detailed results are provided in Section4.3.

Stage 3: Ensemble Prediction. In Stage 3, we perform ensemble prediction using the three models from Stage 2. For each model, we use the example number that yielded relatively good performance in Stage 2 (we uniformly set example=3, as Model 1's accuracy stabilizes at this point, and Models 2 and 3 show emerging diversity in predictions without significant performance drops). We conduct three ensemble runs using weighted majority voting, with weights set to [0.4, 0.3, 0.3]. In each run, one model is assigned the dominant weight of 0.4 in turn, in order to evaluate the effectiveness of different ensemble priorities.

4.3 Results, Tables and Figures

In this section, we report the experimental results of the three models and evidence retrieval stage (Stage 1) introduced in Section 4.2. We evaluate their performance on the train_claims.json and dev_claims.json dataset. For each model, we record separately the Evidence Retrieval F-score and claim classification accuracy. We also present the results of Stage 3 along with the harmonic mean of F and A.

In Table 1, we compare four experimental settings to assess the effects of retriever architecture, reranking, fine-tuning, and preprocessing. Experiment 1, using all-MiniLM-L6-v2 without reranking or data manipulation, serves as our baseline with an F1 of 13.53. Experiment 2 achieves the best performance (F1 = 15.72) by using multi-qa-mpnet-base-dot-v1 with a cross-encoder reranker, confirming that reranking substantially improves recall and overall alignment. In Experiment 3, we expected fine-tuning to improve retrieval, but it instead degraded performance (F1 = 8.39). This is likely due to limited training data (set for 300 claims to save resources) and the use of simple sampling with just three negatives per claim, which may have caused the model to overfit or learn poorly generalizable representations.

Experiment 4 applied a preprocessing pipeline that included lowercasing, punctuation removal, stopword filtering, and lemmatization. However, this also reduced performance (F1 = 11.58), suggesting that aggressive text normalization removed useful semantic cues such as named entities, function words, or syntactic structure, which are important for dense semantic retrieval. Furthermore, multi-qa-mpnet-base-dot-v1 is trained on full-sentence question–answer pairs, so it expects inputs with rich contextual structure. Overly aggressive preprocessing (e.g., stopword removal, punctuation stripping) may disrupt this expected input format and degrade embedding quality.

Overall, our findings in Stage 1 experiments indicate that pretrained models paired with rerankers perform best out-of-the-box, while further modifications like fine-tuning and preprocessing must be carefully tuned to avoid harming performance.

From Table 2, we observe that the best F-score is achieved when $k_t = 50$ and $k_r = 3$. This suggests that increasing the initial retrieval range (k_t) helps include more potentially relevant evidence, compensating for the limitations of dot similarity in capturing semantic relevance. Additionally, applying reranking with a cross-encoder (used in k_r) appears to effectively identify more accurate evidence from the broader pool, leading to better retrieval performance.

From Figure 2. Although nearly half of the claims contain five ground-truth evidence passages, setting a smaller rerank size such as $k_r = 3$ remains effective. This is because a smaller k_r allows the reranking model to focus on the most relevant candidates, reducing noise and improving precision. However, if k_r is too small, it may miss some correct evidence, so a careful balance between precision and recall is necessary.

See Table 3 for a comparison of three prompt-style variants under 1–5 shot settings. The three styles differ only in the level of instruction detail:

- **Simple:** Classify each claim as SUPPORTS, REFUTES, NOT_ENOUGH_INFO, or DISPUTED.

Table 1: Stage 1: Experiment Results on Train Dataset

Exp	Retriever	Reranker	Finetune	Preproc	Precision	Recall	F1
1	all-MiniLM-L6-v2	No	No	No	14.25	14.71	13.53
2	multi-qa-mpnet-base-dot-v1	Yes	No	No	12.86	24.52	**15.72**
3	multi-qa-mpnet-base-dot-v1	Yes	Yes	No	7.12	11.96	8.39
4	multi-qa-mpnet-base-dot-v1	Yes	No	Yes	9.74	16.61	11.58

Table 2: F1 score (%) for different values of k_t (top-k retrieved) and k_r (reranked) on `train_claims.json`.

$K_t \setminus K_r$	3	5	10
10	15.80	15.45	12.06
20	16.44	16.04	13.33
50	16.54	16.45	14.00

Table 3: Accuracy (%) of Different Prompting Styles under Various Few-Shot Settings on `dev_claims.json`

Prompting Style	1-Shot	2-Shot	3-Shot	4-Shot	5-Shot
Simple	42.86	39.61	38.31	37.66	40.91
Middle	40.26	37.01	37.01	35.71	42.21
Complex	42.86	40.91	36.36	38.96	40.26

- **Middle:** Adds "You are an expert fact-checker" to ground the task in a factual-verification role.
- **Complex:** Further instructs the model to "think carefully about how each evidence snippet relates to the claim" before answering.

Contrary to an expectation that more detailed prompts would uniformly drive higher accuracy, the best-performing style actually varies with the number of shots:

- *1-Shot:* Simple and Complex tie for the highest accuracy (42.9 %).
- *2-Shot:* Complex leads at 40.9 %.
- *3-Shot:* Simple peaks at 38.3 %.
- *4-Shot:* Complex again leads at 39.0 %.
- *5-Shot:* Middle achieves the best result (42.2 %).

Across all five settings, the **average** accuracy is highest for Simple prompts (39.9 %), closely followed by Complex (39.7 %) and then Middle (38.4 %). This non-monotonic pattern suggests that:

1. Detailed instructions (Middle, Complex) can help in certain few-shot regimes (e.g. 2- and 4-shot), but do not provide a uniform advantage.
2. The simplest formulation remains highly competitive, particularly when only one or three examples are available.
3. Prompt complexity alone is not a reliable proxy for better generalization; the interplay between example count and instructional detail must be carefully balanced.

Table 4: Comparison of Classification Accuracy (%) between Model 2 (CoT Prompting) and Model 3 (Static Examples) under Few-Shot Settings on `dev_claims.json`

Method	1-Shot	2-Shot	3-Shot	4-Shot	5-Shot
Model 2 (CoT)	44.16	44.16	44.81	35.71	24.03
Model 3 (Static)	44.16	44.16	44.81	35.71	24.03

See Table 4.

Analysis of Identical Outputs from Both Methods. We observe that both the static-example and CoT-prompting approaches yield exactly the same greedy outputs. We attribute this to three main factors:

1. Both methods invoke the same `TinyLLaMA-1.1B-Chat` model with `do_sample=false`, so the model deterministically follows the single highest-probability decoding path.
2. Although the CoT prompt includes "Let's think step by step" and "Reasoning:" cues, the restricted `max_new_tokens` budget prevents the model from generating a full chain of thought, causing it to skip directly to the most likely label.
3. `TinyLLaMA-1.1B-Chat` is not optimized for multi-step reasoning under tight token constraints; it conserves its output "budget" by producing only the final label. We also tested several 3B-parameter open-source models on CoT prompts without consistent improvement.

Analysis of Predominant SUPPORTS Predictions. Nearly all predictions collapse to "SUPPORTS" due to:

1. **Class imbalance.** In our development set (154 examples), SUPPORTS comprises 68/154 (44%) of labels, making it the most frequent class.

2. **Model prior bias.** `TinyLLaMA-1.1B-Chat`'s pretraining likely assigns higher base probability to the token "SUPPORTS" in an open-ended classification setting, especially under greedy decoding and minimal instruction.

Table 5: Accuracy (%) comparison of Model 1, Model 2 (CoT) and Model 3 (Static Examples) across few-shot settings on `train_claims.json`.

Examples	Model 1	Model 2 (CoT)	Model 3
1	56.03	42.18	42.26
2	52.28	42.26	42.26
3	59.04	25.65	39.82
4	63.03	34.53	30.37
5	63.44	11.97	19.54

From the table 5, we see that on the training set, Model 1's accuracy improves as the number of examples increases, while Model 3's accuracy declines. Initially, Model 3 predicts only SUPPORTS, showing high bias. As more examples are added, its output becomes more diverse, but accuracy drops further. This suggests that Model 1 benefits from more examples due to better use of context, while Model 3, lacking advanced strategies, struggles to generalize and interpret claim-evidence relations effectively.

Table 6: Ensemble results with different weight settings for stage3 in training set.

Weights	F1	Acc	HM
(0.4, 0.3, 0.3)	16.44	50.41	24.80
(0.3, 0.4, 0.3)	16.44	34.28	22.22
(0.3, 0.3, 0.4)	16.44	36.56	22.68

As shown in the table 6, when Model 1 is assigned the highest weight (0.4, 0.3, 0.3), the ensemble achieves the best overall performance, with the highest Accuracy and Harmonic Mean (HM). This suggests that Model 1 has the most reliable outputs, and integrating it with other models further improves stability. The weight setting balances accuracy and diversity, indicating that weighted ensemble—giving higher weight to stronger models—is an effective strategy to enhance overall performance.

From Table 7, you can see all three weight settings give almost the same accurac. That's because the three models make very similar predictions and tend to pick SUPPORTS most of the time. In particular, Models 2 and 3 are biased toward SUPPORTS, so no matter how we blend them, the results stay nearly identical.

Table 7: Ensemble results with different weight settings for stage3 on `dev_claims.json`.

Weights	F1	Acc	HM
(0.4, 0.3, 0.3)	24.82	27.27	26.00
(0.3, 0.4, 0.3)	24.73	28.57	26.50
(0.3, 0.3, 0.4)	23.72	27.27	25.37

5 Conclusion

This project presents a modular, memory-efficient fact-checking system that integrates dual encoder retrieval, in-context learning, and ensemble selection across three stages. In Stage 1, our results show that cross-encoder reranking significantly improves retrieval quality when paired with a strong dense retriever. However, aggressive preprocessing and low-resource fine-tuning can negatively impact performance, highlighting the need for alignment between model expectations and data processing. In Stage 2, we find that instruction tuning and prompt formulation significantly influence classification accuracy, and greedy decoding with small models may limit the benefits of chain-of-thought reasoning. Finally, Stage 3 demonstrates that ensemble prediction using harmonic mean optimization provides a balanced trade-off between evidence retrieval and classification performance. Future work should explore more robust fine-tuning strategies, pretrained instruction-tuned LLMs, and more diverse ensemble voting schemes to improve generalization and interpretability in real-world fact-checking scenarios.

Team Contributions

- **Muhammad Md Nasrein:** Stage 1 Experiment, Architecture Design
- **Ni Bin Xin:** Stage 2 and 3 Experiment, Baseline Testing
- **Dong Xiang:** Stage 2 and 3 Experiment, Optimization

Limitations

While our multi-stage fact-checking system achieves promising results, several limitations must be acknowledged. First, our fine-tuning setup in Stage 1 was constrained to only 300 claims and relied on a basic negative sampling strategy with three negatives per claim. This likely limited the model's ability to generalize and may have led to overfitting, as reflected in its degraded performance. Second, our preprocessing pipeline involved aggressive transformations such as stopword removal, lemmatization, and punctuation stripping. Although intended to normalize text, this likely disrupted the input format expected by the dense retriever model (`multi-qa-mpnet-base-dot-v1`), which is pretrained on full-sentence QA pairs and benefits from syntactic richness. Third, our evaluation of retrieval effectiveness and classification accuracy was primarily conducted on the training and development sets, which limits conclusions about generalization to unseen data. Finally, due to resource constraints, we limited our experiments to small-scale open-source models such as `TinyLLaMA-1.1B`, which are not optimized for complex multi-step reasoning and deterministic decoding. These factors may have restricted the diversity and interpretability of generated outputs, especially in chain-of-thought prompting.

References

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.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *Preprint*, arXiv:1901.04085. ArXiv preprint arXiv:1901.04085.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. 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*, pages 809–819.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.