Self-Checker: Plug-and-Play Modules for Fact-Checking with Large Language Models

Miaoran Li[†]
Iowa State University
limr@iastate.edu

Baolin Peng [‡]
Microsoft Research
baolin.peng@microsoft.com

Michel Galley
Microsoft Research
mgalley@microsoft.com

Jianfeng Gao
Microsoft Research
jfgao@microsoft.com

Zhu Zhang
University of Rhode Island
zhuzhang@uri.edu

Abstract

Fact-checking is an essential task in NLP that is commonly utilized to validate the factual accuracy of a piece of text. Previous approaches mainly involve the resource-intensive process of fine-tuning pre-trained language models on specific datasets. In addition, there is a notable gap in datasets that focus on fact-checking texts generated by large language models (LLMs). In this paper, we introduce SELF-CHECKER, a plug-and-play framework that harnesses LLMs for efficient and rapid fact-checking in a few-shot manner. We also present the BINGCHECK dataset, specifically designed for fact-checking texts generated by LLMs. Empirical results demonstrate the potential of SELF-CHECKER in the use of LLMs for fact-checking. Compared to state-of-the-art fine-tuned models, there is still significant room for improvement, indicating that adopting LLMs could be a promising direction for future fact-checking research.

1 Introduction

Fact-checking is an essential task in natural language processing, focusing on evaluating the accuracy of text. The advent of large language models (LLMs), such as ChatGPT, GPT-4 (OpenAI, 2023), and GPT-3 (Brown et al., 2020), has intensified the importance of this task. As LLMs gain widespread use, the risk of generating false information and hallucinating facts becomes a prominent concern. Despite the extensive implicit knowledge in LLMs and their superior ability to generate realistic responses, ensuring the accuracy and truthfulness of their outputs remains a significant challenge.

Researchers have developed methods for fact-checking and subtasks, including claim detection and fact verification (Guo et al., 2022). Traditional fact-checking approaches typically involve fine-tuning LLMs on specific datasets, which can be computationally expensive and time-consuming. The accelerated progress of LLMs has sparked recent exploration into their potential for fact-checking. Pan et al. (2023) proposed ProgramFC which prompts CodeX for reasoning program generation to guide the verification process.

Existing fact-verification datasets (Thorne et al., 2018; Schuster et al., 2021; Petroni et al., 2022; Kamoi et al., 2023) mainly center on verifying claims from Wikipedia, which do not capture the complexity of lengthy and informative texts generated by LLMs. The lack of a suitable fact-checking dataset tailored for LLM generation poses a chal-

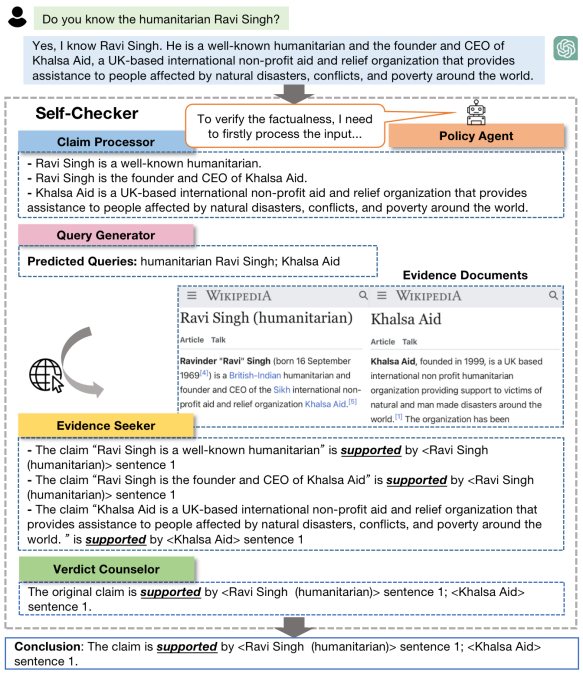

Figure 1: SELF-CHECKER assesses the veracity of LLM generated text by (1) extracting simple claims for verification from the input text, (2) generating search queries for retrieval, (3) selecting evidence sentences, and (4) predicting the final conclusion.

[†] This work was done during an internship at Microsoft Research.

[‡] Currently at Tencent AI Lab. Work done at Microsoft Research.

Findings of the Association for Computational Linguistics: NAACL 2024, pages 163–181

lenge in designing and evaluating frameworks in the evolving landscape of LLMs.

In this paper, we introduce SELF-CHECKER (depicted in Figure 1), a framework comprising plug-and-play LLM modules for automated fact-checking. The primary objective of SELF-CHECKER is to assess the veracity of complex texts (*e.g.*, the response generated by ChatGPT). To achieve this goal, SELF-CHECKER first extracts several simple claims for verification from the input and then predicts search queries for these claims to retrieve documents from a knowledge source (*e.g.*, Wikipedia in this example). After obtaining relevant documents, SELF-CHECKER selects evidence sentences for each claim from the documents and finally returns a veracity prediction (*e.g.*, whether the original claim is supported by evidence). We also construct BINGCHECK dataset, which focuses on verifying the factual accuracy of texts generated by LLMs. We collect interactions between a simulated user and an LLM and hire human annotators to determine the factualness of LLM's responses.

This paper makes the following contributions: (*i*) We introduce SELF-CHECKER to utilize LLMs for automatic fact-checking. (*ii*) We construct BINGCHECK dataset, which facilitates future research on fact-checking in a more realistic setting. (*iii*) We evaluate the effectiveness of SELF-CHECKER on the BINGCHECK dataset and two fact verification datasets. Our experiments show that SELF-CHECKER is capable of generating reasonable results and exhibits considerable potential in the field of fact-checking. While SELF-CHECKER's performance remains below that of state-of-the-art (SOTA) models for fact verification, our approach does not require any fine-tuning and can be applied to any off-the-shelf LLM.

2 SELF-CHECKER Framework

SELF-CHECKER is a framework for fact-checking that is training-free and contains a set of plug-and-play modules—claim processor, query generator, evidence seeker, and verdict counselor. The illustration of SELF-CHECKER is depicted in Figure 2. A comparison of SELF-CHECKER against other related frameworks is provided in Table 1. SELF-CHECKER is designed to assess the factuality of textual inputs and employs a policy agent that strategically plans future actions based on a predefined set of choices. Each module is implemented by prompting an LLM through carefully crafted

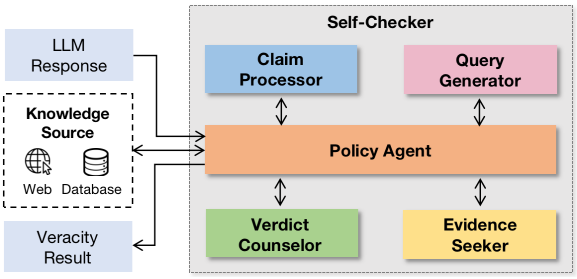

Figure 2: Overview of SELF-CHECKER. The framework consists of four plug-and-play modules: (1) claim processor, (2) query generator, (3) evidence seeker, and (4) verdict counselor.

prompts. Detailed example prompts are provided in Appendix A. This modular approach allows for seamless integration to specific fact-checking requirements but also promotes adaptability in diverse application scenarios.

Policy Agent This module determines the subsequent action of the system from a set of predefined actions. These actions include: (1) calling the claim processor to process the complex input, (2) requesting search queries from the query generator, (3) retrieving relevant passages from a knowledge source based on the generated search queries, (4) utilizing the evidence seeker to extract evidence sentences for a claim from the retrieved passages, (5) requesting the verdict counselor to provide a verdict prediction based on the gathered evidence, and (6) sending the final conclusion to the users.

The policy agent follows the task instruction and learns from in-context examples to select the most appropriate action based on the current state and observations of the framework. The task description includes a comprehensive list of all available modules, along with brief descriptions of their respective functions. In-context examples provide complete processes of fact-checking for sample input text. This decision-making process ensures the efficient execution of the fact-checking process.

Claim Processor The first step in fact-checking is to identify claims for verification from the input text. Traditionally, this task involves classifying whether a sentence constitutes a claim or ranking sentences according to their check-worthiness (Atanasova et al., 2018; Barrón-Cedeño et al., 2020; Zeng et al., 2021). Leveraging the advanced text generation capabilities of LLMs, we redefine the task of obtaining a set of claims to verify as a generation task. Given a text t as in-

Method	Goal	Input	Planning in the process	Knowledge source	Output
Verify-and-Edit (Zhao et al., 2023)	Improve reasoning	CoT reasoning	No	DrQA, Wikipedia, Google search	Revised reasoning
FactTool (Chern et al., 2023)	Evaluate factuality	LLM response	No	Wikipedia, Python, Calculator, Google scholar	Factuality labels
FActScore (Min et al., 2023)	Evaluate factuality	LLM response	No	Wikipedia	Factuality score
FactCheck-GPT (Wang et al., 2023)	Correct factual errors	LLM response	No	Google search	Revised response
Chain-of-Verification (Dhuliawala et al., 2023)	Correct factual errors	LLM response	Generate entire plan	Parametric knowledge	Revised response
SELF-CHECKER (Ours)	Evaluate factuality	LLM response	Generate plan step by step	Bing search	Factuality labels

Table 1: Comparison of related frameworks. SELF-CHECKER aims to provide a factual evaluation of input text, in contrast to FactCheck-GPT and Chain-of-Verification (CoVe), which focus on amending factual inaccuracies in the input text. CoVe revises the input by answering a set of generated verification questions and does not explicitly assess the factuality of the input. While FacTool and FActScore also deliver factual assessment results and FactCheck-GPT can provide intermediate detection results, SELF-CHECKER is distinct in that it utilizes a policy agent to dynamically plan future actions from an array of predetermined options.

put, the claim processor generates a set of claims $\{c_1, c_2, ..., c_m\}$ that are included in t and need to be verified. If a specific claim for verification has been provided, the claim processor can also break it down into a set of simpler claims. Each claim within the set contains a single piece of information, which eases the burden of the subsequent verification process. All generated claims should convey the same information that needs to be verified, as conveyed by the original input. To achieve this generation process, an LLM is prompted with a combination of task instructions, in-context examples, and a piece of text to be examined.

Query Generator In order to verify a claim, it is essential to retrieve pertinent information from an external knowledge source. Given a claim c, the query generator predicts search queries $q = \{q_1, q_2, ..., q_k\}$ for the purpose of information retrieval. These generated queries are then used to obtain relevant passages $\{p_1, p_2, ..., p_k\}$ from a knowledge source. The query generation process is accomplished by prompting an LLM. The prompt for the query generator includes task instructions, in-context examples, and the claim to be verified.

Evidence Seeker The evidence seeker aims to identify evidence sentences for a given claim from the retrieved passages. Given a claim c and the set of retrieved passages $\{p_1, p_2, ..., p_k\}$, the evidence seeker returns a set of selected sentences $\{s_1, s_2, ..., s_n\}$ that indicate the veracity of the claim. To accomplish this process, an LLM is prompted through a specific prompt comprised of task instruction, in-context examples, the claim to be verified, and the retrieved passages.

Verdict Counselor The primary objective of the verdict counselor is to analyze the set of claims that require verification, together with the corresponding evidence sentences for each claim. This module is responsible for predicting the veracity r of the entire set of claims. By examining the provided evidence, the verdict counselor determines the factuality of each claim and assigns an appropriate veracity label, such as *supported*, *partially supported*, or *refuted*. The labels are then aggregated to obtain the final result of the entire set. The veracity labels used by the verdict counselor are predefined, encompassing the degrees of entailment (*e.g.*, supported/partially supported/not supported/refuted). To accomplish this process, an LLM is prompted with specific instructions.

3 The BINGCHECK Dataset

Recent work (Liu et al., 2023) shows that while existing generative search engines powered by LLMs can provide fluent and appear informative responses, they often suffer from hallucination. To alleviate the problem of hallucinations in LLM generation and facilitate fact-checking research in a more realistic setting, we develop the BINGCHECK dataset by human annotation with the assistance of the SELF-CHECKER framework. We aim to annotate texts generated by an LLM that are naturally occurring and fine-grained. We collect responses from LLM to user queries related to various topics, which are relatively long and informative. We process complex response into multiple

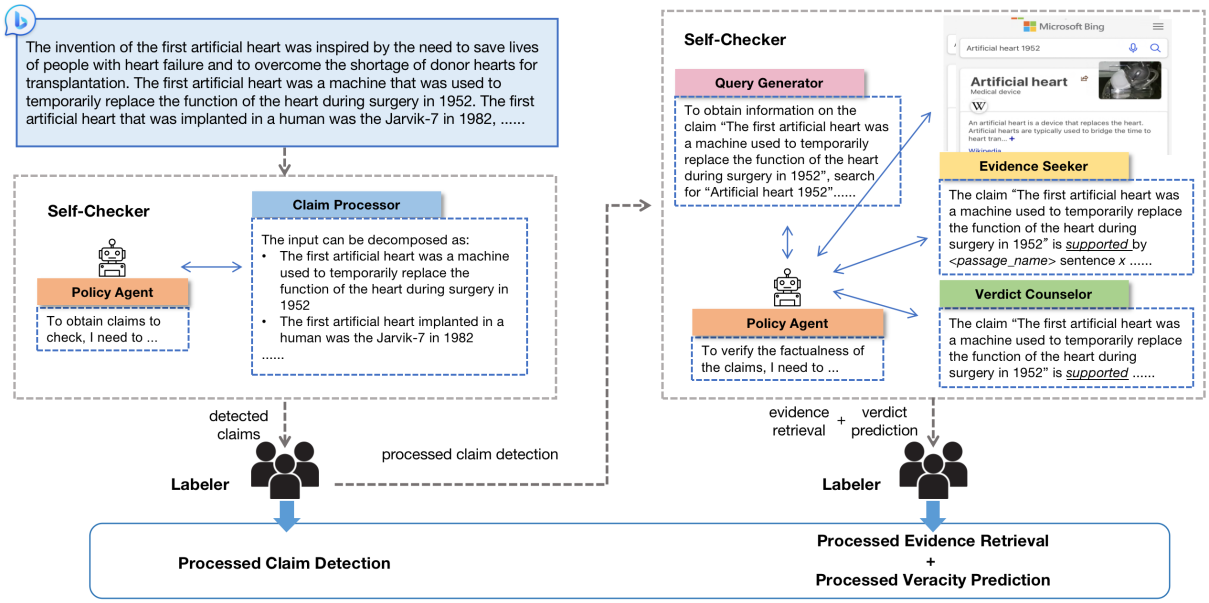

Figure 3: Illustration of BINGCHECK dataset construction. The initial claim detection results are obtained using SELF-CHECKER, and human annotators verify and refine these automatic results. Processed claims are entered into SELF-CHECKER for fact verification data generation, and the outputs are further validated by human workers.

simple claims that are worth-checking and the provide fact-checking information for both response level and claim level.

3.1 Dataset Construction

3.1.1 Base Data Collection

To collect responses to various user queries generated by an LLM, we adopt ChatGPT to simulate a curious user and gather responses generated by Bing Chat[1]. We prompt ChatGPT with a user persona characterized by curiosity and an inclination to ask questions on various topics and collect 396 interaction instances between the simulated user and Bing Chat. The responses generated by Bing Chat serve as the input text to be verified.

3.1.2 Data Annotation

After collecting the base data on the interaction between the simulated user and Bing Chat, we hired human workers on Amazon Mechanical Turk to annotate the data. We aim to autogressively collect annotated data for three subtasks: (1) claim detection, (2) evidence retrieval, and (3) veracity prediction. To ensure the quality of data annotation, we have launched onboarding tasks to select proficient workers. Onboarding tasks mirror main tasks but are less demanding. Only qualified workers who pass the onboarding task access the primary task with higher rewards. Each record in BINGCHECK is then labeled by a qualified worker.

Considering the potential challenges and time constraints associated with human annotation, we adopt the SELF-CHECKER framework to assist in the following annotation process. The main idea is that for each subtask, we first utilize the SELF-CHECKER framework to generate candidate solutions to a subtask and then require human annotators to validate and correct the candidate solutions. The processed solutions are used to generated candidate solutions to the next subtask. The human-processed data are collected in BINGCHECK. The data collection process is depicted in Figure 3. The instruction for human annotation and an example of annotated data are shown in Appendix B.

Claim Detection Using the SELF-CHECKER framework, particularly the claim processor module, we generate a set of claims for verification for each input. Human workers then assess and correct the automatically labeled data. Workers receive a Bing Chat response and a set of claims extracted by SELF-CHECKER. Their task involves selecting all claims in the response that necessitate verification from the provided set and filling in any missing claims requiring verification but not included in the given set.

[1]It is named as Bing Chat when we collected the data. It has been updated to Microsoft Copilot now. The implementation is based on https://github.com/acheong08/EdgeGPT

Statistic	Response	Extracted Claim
Total number	396	3840
Average length	391.5	26.3
Number of evidence sentences	55.0	6.2
Number of claims per response	9.7	-

Table 2: Statistics of the BINGCHECK dataset. The "Response" column stands for raw response generated by BingChat, and "Extracted Claim" represents a claim extracted from a response that needs to be verified. The number of evidence sentences is computed only on responses/claims with SUPPORTED, PARTIALLY SUPPORTED, REFUTED labels.

Evidence Retrieval and Veracity Prediction Claims processed by workers are inputted into the SELF-CHECKER framework, integrating the query generator, evidence seeker, and verdict counselor modules. For each claim, SELF-CHECKER predicts a search query, retrieves relevant passages from a certain knowledge source,[2], selects evidence sentences, and predicts the candidate veracity label.

We consider four veracity labels: SUPPORTED, PARTIALLY SUPPORTED, REFUTED, NOT SUPPORTED. A claim is refuted if any evidence sentence contradicts it. A claim is supported if there are no refuting sentences and at least one sentence supporting it. A claim is partially supported when there are sentences that contribute to the credibility of a portion of the claim but do not fully establish its truth or validity. A claim is not supported if there are no sentences that refute, support, or partially support the claim.

The automatic results of evidence retrieval and claim verification are provided to workers. Their task involves reviewing the claim along with each automatically selected evidence sentence, selecting all sentences relevant to verifying the claim's factuality. Finally, the workers determine the verdict results based on their selection.

3.2 Statistics

Table 2 presents the overall statistics for the BINGCHECK dataset. The original responses generated by Bing Chat have an average length of 391.5 tokens and can be decomposed into an average of 9.7 claims for verification. The dataset contains more than 3800 claims. For claims that are refuted, supported, or partially supported, there are approximately 6 evidence sentences on average.

Table 3 presents a comparative analysis of BingCheck against established datasets in the fact-checking field. Our dataset is characterized by its considerably longer responses compared to those found in the existing datasets. This significant increase in response length suggests that BingCheck can provide a more complex and extensive framework for assessing factuality. Furthermore, this increased length underscores the alignment of our dataset with real-world scenarios, wherein responses to complex or broad inquiries posed to LLMs are typically extensive and detailed, thereby making the factuality evaluation more challenging.

3.3 Dataset Quality Evaluation

To evaluate the quality of the annotated data, we have hired Amazon Mechanical Turk workers to perform annotation review tasks. For each annotated record, we have employed three workers to evaluate it. Each worker answers a series of single-choice questions to assess the quality of the annotation. To evaluate the quality of claim detection, a worker is presented with an original response and an annotated list of claims. The workers need to determine whether all listed claims need verification and whether all claims in the response that require verification are included in the given set of claims. To assess the quality of annotations for evidence retrieval and veracity prediction, a worker is presented with a claim and a list of evidence sentences. A worker first determines whether all evidence sentences are relevant for verifying the claim's factualness. Then the worker determines whether the assigned label is correct. We use a majority vote to aggregate the evaluation results.

In terms of claim detection, among all 396 records, the extracted claims in 381 records are deemed comprehensive and verifiable. However, there are 15 records where the claim detection is either missing or contains claims that do not require verification. Regarding evidence retrieval and veracity prediction, we have a total of 3840 extracted claims. Evaluators have found that 94% of these claims have appropriate evidence sentences. In the case of the remaining claims, there may be redundant and irrelevant sentences within the selected evidence. For verdict prediction, 96% claims have been considered to be accurately assigned with appropriate labels based on the annotated evidence. There may be some level of noise in the human evaluation results. Nevertheless, this evaluation process provides an estimation of dataset quality

[2]In our implementation, we utilized the Bing search engine and retrieved three passages for each claim.

Dataset	Input		Claim granularity	Knowledge Source	Evidence provided	Task	Scenario
	Length	Generated by					
Fever (Thorne et al., 2018)	7.3	Human	fact	Wikipedia	Yes	Fact verification	Wikipedia claim
WiCE (Kamoi et al., 2023)	24.2	Human	Fact	Wikipedia	Yes	Entailment classification	Wikipedia claim
FactTool (Chern et al., 2023)	76.3	ChatGPT	Fact	Wikipedia, Python, Calculator, Google scholar	Yes	Fact checking	QA, Code, Math, Literature review
HaluEval (Li et al., 2023)	82.0	ChatGPT	Response	Parametric knowledge	No	Fact checking	QA, Dialog, Summary
FELM (Chen et al., 2023)	89.1	ChatGPT	Segment	Google search	Yes	Fact checking	World knowledge, Science, Math, Recommendation
FActScore (Min et al., 2023)	154.5	InstructGPT, ChatGPT, PerplexityAI	Response	Wikipedia	No	Fact checking	Biography generation, Long-form response
FactCheck-GPT (Wang et al., 2023)	95.8	ChatGPT, GPT4	Response	Google search	Yes	Fact checking and error correction	QA
Chain-of-Verification (Dhuliawala et al., 2023)	-	Llama-65B	-	Search engine	-	Factual error correction	QA
BingCheck (Ours)	391.5	Bing Chat	Response	Bing search	Yes	Fact checking	Long-form response, QA

Table 3: Comparison of factuality evaluation datasets. The "Scenario" column describes the tasks used to gather the initial responses. The critical point of differentiation for our dataset is the significantly greater average response length, which is considerably longer than those in the datasets we have compared it with.

and offers valuable insights for further checks and improvements in data annotation.

4 Experiments

4.1 Datasets

We evaluate the performance of the Self-Checker framework for the fact-checking task on the BingCheck dataset. Additionally, we assess its efficiency in performing fact verification using the FEVER dataset (Thorne et al., 2018) and text entailment using the WiCE dataset (Kamoi et al., 2023).

BingCheck Dataset The fact-checking process of LLM response in the BingCheck dataset involves four subtasks: (1) Claim detection: Given a long paragraph t, models are required to generate a set of claims $\{c_1, c_2, ..., c_m\}$ that require evidence or proof to support their accuracy or truthfulness. (2) Document retrieval: Given a claim c, models are expected to predict search queries $\{q_1, q_2, ..., q_k\}$ to retrieve relevant articles from a knowledge source. (3) Sentence retrieval: Given a claim c and relevant passages $\{p_1, p_2, ..., p_k\}$, models are required to select evidence sentences $\{s_1, ..., s_n\}$ from the articles. These evidence sentences can either (partially) support or refute the claim, depending on the veracity label design. (4) Verdict prediction: Given a claim c and the evidence sentences $\{e_1, ..., e_n\}$, models are required to predict the veracity label. The fact-checking process requires the claim processor, query generator, evidence seeker, and verdict counselor modules.

FEVER Dataset In the FEVER (Thorne et al., 2018) dataset, claims consist of a single piece of information and do not require further decomposition. The verification of a claim in FEVER involves document retrieval, sentence retrieval and verdict prediction. The FEVER dataset uses three identification labels: Supported, Refuted, and NotEnoughInfo. A claim is verified as NotEnoughInfo if there is insufficient information in Wikipedia to support or refute the claim, either because the claim is too general or too detailed. The dataset provides the names of evidence Wikipedia passages and the indices of evidence sentences. In the verification process, the names of evidence articles serve as search queries. To verify a claim in the FEVER dataset, the Self-Checker framework adopts query generator, evidence seeker, and verdict counselor. We follow the experiment setting in the previous research (Zhao et al., 2023) and use the same subset of Fever.

WiCE Dataset The WiCE dataset is specifically designed for verifying Wikipedia citations and consists of claims grounded in cited articles from Wikipedia. Unlike the FEVER dataset, the claims in WiCE contain multiple pieces of information. The verification process in WiCE involves claim detection, sentence retrieval, and verdict prediction. Complex claims in WiCE are decomposed into sim-

pler subclaims. Verifying claims in WiCE primarily entails sentence retrieval for the cited articles and subsequent verdict prediction. The veracity labels in WiCE include SUPPORTED, PARTIALLYSUPPORTED, and NOTSUPPORTED. A claim is classified as PARTIALLYSUPPORTED if some tokens within the claim are not supported by any evidence sentence. The prediction results are collected at subclaim levels. The veracity label of the original claim is set to SUPPORTED or NOTSUPPORTED, depending on whether all subclaims are supported or not supported. Otherwise, the original claim is considered PARTIALLYSUPPORTED. To verify a claim in the WiCE dataset, the SELF-CHECKER framework adopts claim processor, evidence seeker, and verdict counselor modules.

4.2 Experimental Setup

Implementation All modules in the SELF-CHECKER are implemented using OpenAI GPT-3.5 (text-davinci-003) API with temperature 0.2. The prompt for policy agent consists of three examples due to the length constraint. The prompts for claim processor, query generator, evidence seeker, and verdict counselor contain fifteen examples. As for the knowledge source, we employ Bing search engine for BingCheck and Wikipedia for FEVER. Up to three retrieved passages are considered for further evidence selection. In the implementation, we stored FEVER preprocessed Wikipedia passages in a database. The retrieval mechanism automatically incorporates passages whose titles precisely match the generated search query or exhibit partial alignment with the predicted search query.

Evaluation Metrics We report label accuracy and F1 score for evidence retrieval, which is computed between all predicted sentences and the golden evidence sentences for claims requiring evidence. Consistent with baseline studies (Kamoi et al., 2023; Thorne et al., 2018), we present the F1 score for verdict prediction on the WiCE dataset and the FEVER score for results on the FEVER dataset. The FEVER score is the strict accuracy with the requirement of providing correct evidence for the SUPPORTED/REFUTED predictions.

Baselines We evaluate SELF-CHECKER against various methods. Standard prompting directly predicts verdict labels based on input claims, while Chain-of-thought prompting (Wei et al., 2022) generates explanations before making predictions. ReAct (Yao et al., 2023) follows a reason-and-act framework with an external knowledge source[3]. The setup of the knowledge source is similar to that in SELF-CHECKER. We also compare with a related method Verify-and-Edit (Zhao et al., 2023) on Fever dataset. These prompt-based methods are implemented using the OpenAI GPT-3.5 API.

In addition, we compare our approach to the initial baseline model (Thorne et al., 2018) and the state-of-the-art (SOTA) model BEVERS (DeHaven and Scott, 2023) on the FEVER dataset. The baseline model consists of a DrQA (Chen et al., 2017) document retrieval module, a DrQA-based sentence retrieval module, and an entailment module based on decomposable attention (Parikh et al., 2016). The SOTA model adopts BERT for evidence retrieval and claim verification, along with meticulous hyperparameter tuning. For the WiCE dataset, we include the initial baseline model (Kamoi et al., 2023), implemented by fine-tuning T5-3B (Raffel et al., 2020) on WiCE.

4.3 Main Results

Evaluation Results on BINGCHECK Dataset The evaluation results on BINGCHECK are presented in Table 4. We observe the inherent challenge LLMs face when determining the factualness of complex paragraphs based solely on pre-trained parametric knowledge. It is notable that LLMs prompted with standard and chain-of-thought prompts tend to align with the input, tending to recognize it as supported information. The integration of external knowledge contributes to the improvements in fact-checking. However, a performance gap persists between baseline models and the proposed framework, which underscores the importance of incorporating modules capable of decomposing complex paragraphs into simpler claims, conducting explicit analysis of retrieved passages, and predicting verdicts. Furthermore, the availability of intermediate results from the fact-checking process enhances our ability to identify performance bottlenecks within SELF-CHECKER, making it possible to guide further improvements. Despite the introduction of SELF-CHECKER, there are limitations in achieving optimal results on BINGCHECK, highlighting the inherent difficulty in fact-checking LLM-generated content and prompting further exploration.

[3]ReAct is not evaluated on the WiCE dataset as the knowledge retrieval is not included in the verification process for claims in WiCE.

Model	Accuracy	Evidence Retrieval		
		F1	Precision	Recall
Standard Prompt	19.4	-	-	-
Chain-of-Thought	15.7	-	-	-
ReAct (Yao et al., 2023)	21.0	-	-	-
SELF-CHECKER	63.4	45.0	30.5	86.1

Table 4: Evaluation results on BINGCHECK. The accuracy is computed on the response level.

Evaluation Results on FEVER Dataset The evaluation results on the FEVER dataset are presented in Table 5. Compared to prompt-based baselines, SELF-CHECKER improves verification accuracy with explicit evidence retrieval results. Comparing the performance of the baselines and SELF-CHECKER, we observe that LLMs possess a robust capacity to learn from few examples and perform various tasks, including query generation, retrieval and verdict prediction. However, the significant performance gap between the SOTA model and the SELF-CHECKER highlights the need to improve the efficiency of the SELF-CHECKER.

Evaluation Results on WiCE Dataset The evaluation results for the WiCE dataset are in Table 6. The F1 score for label prediction is quite low for the LLM with standard prompting, as it tends to predict the supported claim as partially or not supported. In line with earlier findings, SELF-CHECKER demonstrates superior efficiency compared to the prompt-based baselines. A noticeable performance gap emerges when comparing SELF-CHECKER with the model fine-tuned on the WiCE dataset. Specifically, in evidence retrieval, evidence seeker tends to overlook evidence in the passages, highlighting a potential bottleneck in overall performance.

4.4 Ablation Study

To assess the impact of each module on overall performance, we conduct an ablation study on three datasets. The evaluation results on BINGCHECK are shown in Table 7. The first row reflects end-to-end fact-checking performance, encompassing claim detection, document retrieval, sentence retrieval, and verdict prediction. When comparing the first and second rows, we note that providing golden claims results in improvements across all metrics. The marginal difference between results with and without golden documents suggests the low-temperature setting of the API in the query generator module ensures stable search query generation, with retrieval results for a fixed query exhibiting consistency. Even with golden evidence sentences, the label accuracy at the response level does not exceed 70, indicating potential for further enhancements in the verdict counselor module to improve the accuracy of veracity prediction. In terms of evidence retrieval performance, it is unsurprising to observe an inclination to over-select more evidence sentences. This behavior stems from the dataset construction process, where human workers filter evidence sentences selected by SELF-CHECKER, removing less relevant ones.

Analyzing the incorrect predictions with golden evidence sentences, we observe a tendency in SELF-CHECKER to be overly optimistic, classifying claims that are only partially supported as fully supported. For instance, the claim "Brain virus was released on 19 January 1986 by two brothers from Pakistan, Basit and Amjad Farooq Alvi." is partially supported by the evidence sentence "In 1986, Brain was developed by the Pakistani brothers Basit and Amjad Farooq Alvi, who were annoyed at having their heart monitoring software copied for free." However, SELF-CHECKER overlooks the lack of mention of the exact release date of the Brain virus and predicts the claim as supported based on the evidence.

5 Related Work

The framework for automated fact-checking involves claim detection and factual verification (Zeng et al., 2021; Guo et al., 2022). Claim detection identifies statements needing verification, while factual verification includes evidence retrieval and assessment of claim validity.

Claim detection has been approached as a binary classification task, determining if a sentence represents a claim (Hassan et al., 2017), or as a ranking task, ordering sentences based on their check-worthiness (Jaradat et al., 2018).

Fact verification requires models to assess the veracity of a given claim by examining evidence information. FEVER dataset (Thorne et al., 2018) is one of the most popular datasets in this area, and fueled the development of fact verification models (Soleimani et al., 2020; Jiang et al., 2021; Krishna et al., 2022). The fact verification in FEVER dataset consists of document retrieval, sentence selection, and verdict prediction.

The Vitamin C dataset (Schuster et al., 2021) is proposed for a contrastive fact verification paradigm which requires models to be sensi-

Model	Fine-tuning	FEVER Score	Accuracy	Evidence Retrieval		
				F1	Precision	Recall
Standard Prompt	✗	-	49.9	-	-	-
Chain-of-Thought	✗	-	51.8	-	-	-
ReAct	✗	-	51.4	-	-	-
Verify-and-Edit	✗	-	53.9	-	-	-
SELF-CHECKER	✗	47.9	56.7	47.5	75.3	34.7
DrQA (Thorne et al., 2018)	✓	31.9	50.9	17.5	10.8	45.9
BEVERS (DeHaven and Scott, 2023)	✓	77.7	80.2	-	-	-

Table 5: Evaluation results on FEVER dataset. "Fine-tuning" stands for whether the training procedure is required. Verify-an-Edit is experimented with three different knowledge sources (Zhao et al., 2023). We compare with the highest accuracy obtained by using the Google search engine as a knowledge source.

Model	Fine-tuning	F1	Accuracy	Evidence Retrieval		
				F1	Precision	Recall
Standard Prompt	✗	9.0	65.9	-	-	-
Chain-of-Thought	✗	36.7	50.0	-	-	-
SELF-CHECKER	✗	47.7	71.5	60.5	71.4	52.5
T5-3B (Kamoi et al., 2023)	✓	65.3	77.1	67.4	65.0	81.7

Table 6: Evaluation results on WiCE test set. "Fine-tuning" stands for whether the training procedure is required. Note that we compare with T5-3B model finetuned on WiCE dataset (Kamoi et al., 2023).

Golden Claims	Golden Evidence		Accuracy	Evidence Retrieval		
	Document	Sentence		F1	Precision	Recall
✗	✗	✗	63.4	45.0	30.5	86.1
✓	✗	✗	64.3	48.8	32.7	96.5
✓	✓	✗	64.3	49.0	32.8	97.0
✓	✓	✓	67.2	-	-	-

Table 7: Ablation results on BINGCHECK. "Golden Claims" indicates whether the golden claims are given. "Golden Evidence" indicates whether the golden documents and sentences are provided.

tive to changes in evidence and claims. The WAFER dataset (Petroni et al., 2022) contains instances from Wikipedia inline citations. The WiCE dataset (Kamoi et al., 2023) provided fine-grained annotation of supporting evidence and non-supported tokens in claims.

While many work focused on verifying claims against raw text evidence, other recent datasets cover verification against various evidence, such as table (Chen et al., 2019; Gupta et al., 2020; Akhtar et al., 2022), knowledge graph (Zhu et al., 2021; Vedula and Parthasarathy, 2021; Kim et al., 2023) and other multimodal evidence (Alam et al., 2022).

Factual error correction is a task closely related to fact-checking. After assessing the factualness of claims within the input text, a subsequent step is addressing any inaccuracies to improve factual integrity. Recent studies have explored methods for refining the factualness of text outputs by leveraging retrieved evidence (Thorne and Vlachos, 2021; Iv et al., 2022; Huang et al., 2023). In addition to approaches specialized in correcting factual errors, some recent frameworks first assess the factualness of its initial generation and then amend any detected inaccuracies to enhance the overall veracity of the generation (Wang et al., 2023; Dhuliawala et al., 2023; Fatahi Bayat et al., 2023).

6 Conclusion and Future Work

We present SELF-CHECKER, a framework for automated fact-checking with plug-and-play modules implemented through prompting LLMs. Additionally, we introduce the BINGCHECK dataset, which serves as a valuable resource for future research in fact-checking of LLM-generated responses. Experimental results demonstrate the significant potential of SELF-CHECKER in the fact-checking task.

In future work, a key direction to explore is to enhance the efficiency of SELF-CHECKER. One potential avenue is the incorporation of additional working memory to accelerate the verification process by using past information. Furthermore, investigating more efficient strategies for utilizing LLMs in each subtask of fact-checking holds promise for optimizing performance.

Limitations

One limitation of SELF-CHECKER is its inability to account for information updates. If there is out-of-date information that contradicts a claim, SELF-CHECKER may classify the claim as refuted even if it is actually supported by the most up-to-date information. This limitation arises due to the mixed and unrefined sources of information used by SELF-CHECKER during the fact-checking process. SELF-CHECKER does not contain a module to postprocess and filter the retrieved articles. Another limitation of SELF-CHECKER is its high computational cost due to the involvement of multiple chained LLM calls in the process of fact-checking. To ensure the reliability of predictions, we adopt the majority voting approach by running evidence seeker and verdict counselor multiple times. Although this approach can improve accuracy and stability, it may result in slower response times. However, we anticipate that this limitation can be mitigated in the future with the advancement of more efficient and accessible LLMs. In addition, we will explore providing options to achieve a balance between accuracy and waiting time, allowing users to make informed trade-offs based on their specific requirements. Another limitation is the sensitivity of SELF-CHECKER to prompts. In our preliminary experiments, we have observed variations in performance when using different prompts. Enhancing the robustness of LLMs to prompts is an avenue for future exploration, aiming to improve the reliability and consistency of SELF-CHECKER. Furthermore, the current prompts are manually designed, which may be heuristic in nature. We consider investigating automated methods for selecting in-context learning examples and generating strong prompts in the future work. Additionally, the selection of hyperparameters in SELF-CHECKER currently relies on heuristics. Exploring more efficient automated approaches for hyperparameter tuning could improve the overall efficiency of the framework.

A potential limitation of the BINGCHECK dataset is the potential bias during annotation. The classification of the veracity of a claim can be subjective. It is important to consider this factor when interpreting and utilizing the BINGCHECK dataset for research purposes.

Ethics Statement

In this work, we focus on utilizing SELF-CHECKER to tackle the problem of hallucinations in the generation results of LLMs. However, it is important to acknowledge that LLMs' generation can also exhibit other potential issues, including the production of offensive and harmful content. Currently, SELF-CHECKER does not address these problems. To mitigate these concerns, future work on SELF-CHECKER could incorporate a dedicated module specifically designed to detect and remove offensive and harmful content.

References

Mubashara Akhtar, Oana Cocarascu, and Elena Simperl. 2022. Pubhealthtab: A public health table-based dataset for evidence-based fact checking. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1–16.

Firoj Alam, Stefano Cresci, Tanmoy Chakraborty, Fabrizio Silvestri, Dimiter Dimitrov, Giovanni Da San Martino, Shaden Shaar, Hamed Firooz, and Preslav Nakov. 2022. A survey on multimodal disinformation detection. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6625–6643.

Pepa Atanasova, Alberto Barron-Cedeno, Tamer Elsayed, Reem Suwaileh, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the clef-2018 checkthat! lab on automatic identification and verification of political claims. task 1: Check-worthiness. *arXiv preprint arXiv:1808.05542*.

Alberto Barrón-Cedeño, Tamer Elsayed, Preslav Nakov, Giovanni Da San Martino, Maram Hasanain, Reem Suwaileh, Fatima Haouari, Nikolay Babulkov, Bayan Hamdan, Alex Nikolov, et al. 2020. Overview of checkthat! 2020: Automatic identification and verification of claims in social media. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22–25, 2020, Proceedings 11*, pages 215–236. Springer.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Shiqi Chen, Yiran Zhao, Jinghan Zhang, I-Chun Chern, Siyang Gao, Pengfei Liu, and Junxian He. 2023. FELM: Benchmarking factuality evaluation of large

language models. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. 2023. Factool: Factuality detection in generative ai – a tool augmented framework for multi-task and multi-domain scenarios.

Mitchell DeHaven and Stephen Scott. 2023. Bevers: A general, simple, and performant framework for automatic fact verification. *arXiv preprint arXiv:2303.16974*.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models.

Farima Fatahi Bayat, Kun Qian, Benjamin Han, Yisi Sang, Anton Belyy, Samira Khorshidi, Fei Wu, Ihab Ilyas, and Yunyao Li. 2023. FLEEK: Factual error detection and correction with evidence retrieved from external knowledge. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 124–130, Singapore. Association for Computational Linguistics.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. Infotabs: Inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324.

Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1803–1812.

Kung-Hsiang Huang, Hou Pong Chan, and Heng Ji. 2023. Zero-shot faithful factual error correction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5660–5676, Toronto, Canada. Association for Computational Linguistics.

Robert Iv, Alexandre Passos, Sameer Singh, and Ming-Wei Chang. 2022. FRUIT: Faithfully reflecting updated information in text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3670–3686, Seattle, United States. Association for Computational Linguistics.

Israa Jaradat, Pepa Gencheva, Alberto Barrón-Cedeño, Lluís Màrquez, and Preslav Nakov. 2018. ClaimRank: Detecting check-worthy claims in Arabic and English. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 26–30, New Orleans, Louisiana. Association for Computational Linguistics.

Kelvin Jiang, Ronak Pradeep, and Jimmy Lin. 2021. Exploring listwise evidence reasoning with t5 for fact verification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 402–410.

Ryo Kamoi, Tanya Goyal, Juan Diego Rodriguez, and Greg Durrett. 2023. Wice: Real-world entailment for claims in wikipedia. *arXiv preprint arXiv:2303.01432*.

Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. 2023. FactKG: Fact verification via reasoning on knowledge graphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16190–16206, Toronto, Canada. Association for Computational Linguistics.

Amrith Krishna, Sebastian Riedel, and Andreas Vlachos. 2022. Proofver: Natural logic theorem proving for fact verification. *Transactions of the Association for Computational Linguistics*, 10:1013–1030.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. HaluEval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464, Singapore. Association for Computational Linguistics.

Nelson F Liu, Tianyi Zhang, and Percy Liang. 2023. Evaluating verifiability in generative search engines. *arXiv preprint arXiv:2304.09848*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. 2023. Fact-checking complex claims with

program-guided reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6981–7004, Toronto, Canada. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Fabio Petroni, Samuel Broscheit, Aleksandra Piktus, Patrick Lewis, Gautier Izacard, Lucas Hosseini, Jane Dwivedi-Yu, Maria Lomeli, Timo Schick, Pierre-Emmanuel Mazaré, et al. 2022. Improving wikipedia verifiability with ai. *arXiv preprint arXiv:2207.06220*.

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*, 21:1–67.

Tal Schuster, Adam Fisch, and Regina Barzilay. 2021. Get your vitamin c! robust fact verification with contrastive evidence. *arXiv preprint arXiv:2103.08541*.

Amir Soleimani, Christof Monz, and Marcel Worring. 2020. Bert for evidence retrieval and claim verification. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42*, pages 359–366. Springer.

James Thorne and Andreas Vlachos. 2021. Evidence-based factual error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3298–3309, Online. Association for Computational Linguistics.

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, Volume 1 (Long Papers)*, pages 809–819.

Nikhita Vedula and Srinivasan Parthasarathy. 2021. Face-keg: Fact checking explained using knowledge graphs. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 526–534, New York, NY, USA. Association for Computing Machinery.

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. 2023. Factcheck-gpt: End-to-end fine-grained document-level fact-checking and correction of llm output.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Xia Zeng, Amani S. Abumansour, and Arkaitz Zubiaga. 2021. Automated Fact-Checking: A Survey. ArXiv:2109.11427 [cs].

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5823–5840, Toronto, Canada. Association for Computational Linguistics.

Biru Zhu, Xingyao Zhang, Ming Gu, and Yangdong Deng. 2021. Knowledge enhanced fact checking and verification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3132–3143.

A Example Prompts for SELF-CHECKER

The example prompts for modules in SELF-CHECKER are shown in Figure 4, 5, 6, 7, 8.

B BINGCHECK Dataset

B.1 Human Annotation Instruction

We collected human annotated data for BINGCHECK in two steps. The design of annotation for claim decomposition is shown in Figure 9. The design of annotation for evidence retrieval and veracity prediction is shown in Figure 10.

B.2 Data Format in BINGCHECK

A record in BINGCHECK contains user query, original LLM response, and fact-checking annotation. The fact-checking annotation involves claims to verify, search queries, search results, selected evidence, and verdict labels. Figure 11 shows an annotated record example.

Try your best to determine if the given input response is factually accurate.

<tool introduction>

Use the following format:

Response: the response of language model to the user query. you must verify the factual accuracy of the response. If the input is to long, summarize it without changing factualness.
Thought: you should always realize what you have known and think about what to do and which tool to use.
Action: the action to take, should be one of [actions]
Action Input: the input to the action, must follow instructions of tools
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I can give an answer based on the evidence
Final Answer: should be in the form: supported, partially supported, not supported, refuted

<in-context examples>

Begin!

<text to verify>

Figure 4: Example prompt for the policy agent.

You and your partners are on a mission to fact-check a claim that may contain multiple subclaims that need to be verified. A sentence that needs to be verified is any statement or assertion that requires evidence or proof to support its accuracy or truthfulness. For example, “Titanic was first released in 1997” necessitates verification of the accuracy of its release date, whereas a claim like "Water is wet" does not warrant verification. Each subclaim is a simple, complete sentence with single point to be verified. Imagine yourself as an expert in processing complex paragraphs and extracting subclaims. Your task is to extract clear, unambiguous subclaims to check from the input paragraph, avoiding vague references like ’he,’ ’she,’ ’it,’ or ’this,’ and using complete names.

To illustrate the task, here are some examples:
<in-context examples>

Now, let’s return to your task. You are given the following input paragraph, please extract all subclaims that need to be checked.

Input: <input>
Subclaims: <extracted claims>

Figure 5: Example prompt for the claim processor module. <Extracted claims> is the expected output of the LLM for claim processor.

You and your partners are on a mission to fact-check a paragraph. Subclaims requiring verification have been extracted from the paragraph. Imagine yourself as an internet research expert. Your task is to generate a search query for each subclaim to find relevant information for fact-checking. You will be provided with the context of a claim and the specific claim for which you should create a search query.

To illustrate the task, here are some examples:
<in-context examples>

Now, let’s return to your task. You are given the following claim and its context, please predict the most appropriate search query for it.

Context: <original input text>
Claim: <claim to verify>
Query: <predicted search queries>

Figure 6: Example prompt for the query generator module. <Predicted search queries> is the expected output of the LLM for query generator.

You and your partners are on a mission to fact-check a claim. Your mission is to verify a claim's factual accuracy. As experts in reading comprehension, you'll receive a claim and a passage. You should first read the claim and the passage carefully. Make sure you understand what information you are looking for. Then select sentences that either support, partially support, or refute the claim. A sentence supports the claim if it provides evidence for all statements in the claim. A sentence partially supports the claim if it confirms some details but not all. A sentence refutes the claim if it contradicts any statement in the claim. Exercise caution in your selection and judgment, avoiding overstatement. Choose the most relevant evidence and refrain from including noisy information. Base decisions solely on provided information without implying additional details.

To illustrate the task, here are some examples:
<in-context examples>

Now, let's focus on your task. You are given a claim and a passage. Please read the passage carefully and copy sentences that contain information supporting or refuting the claim.

Claim: <claim to verify>
Passage: <passage>
Evidence: <selected evidence>

Figure 7: Example prompt for the evidence seeker. <Selected evidence> is the expected output of the LLM for evidence seeker.

You and your partners are on a mission to fact-check a claim. Your mission is to verify the factual accuracy of a claim using provided evidence. Your partners have collected evidence, and your expertise lies in assessing the claim's factualness based on this evidence. You are required to determine whether the claim is supported, refuted, or lacks sufficient information based on the provided evidence. The evidence supports the claim if it confirms all statements and details in the claim. The evidence refutes the claim if it contradicts or disproves any statement in the claim. 'Not enough info' applies when the evidence lacks sufficient data, details, or reasoning to support or refute the claim. Even if the evidence supports part of the claim, it should be considered "not enough info" if there is any detail or statement in the claim that cannot be confirmed by the evidence. Please exercise caution in making judgments and avoid overstatement. Base decisions solely on the provided information without implying additional details.

Here are examples to illustrate the task:
<in-context examples>

Claim: <claim to verify>
Evidence: <selected evidence>
Analysis: <verdict prediction>

Figure 8: Example prompt for the verdict counselor. <Verdict prediction> is the expected output of the LLM for verdict counselor.

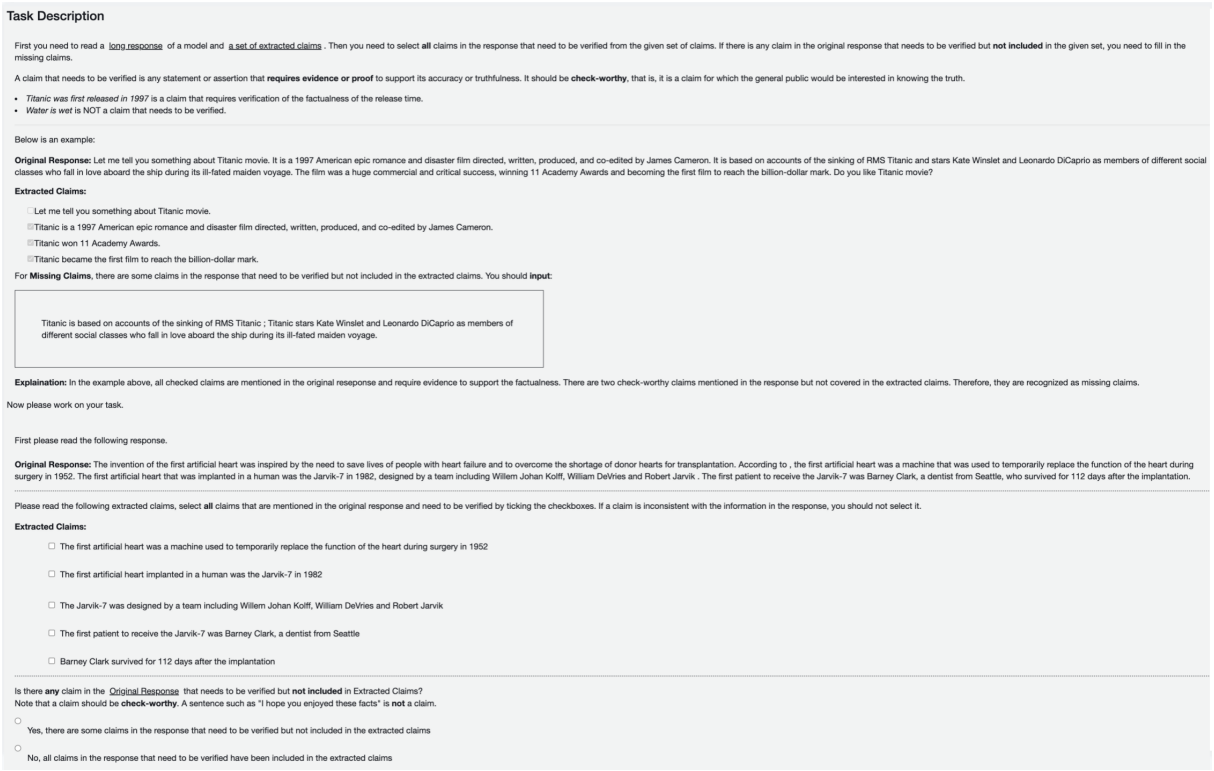

Figure 9: Design of human annotation for claim detection

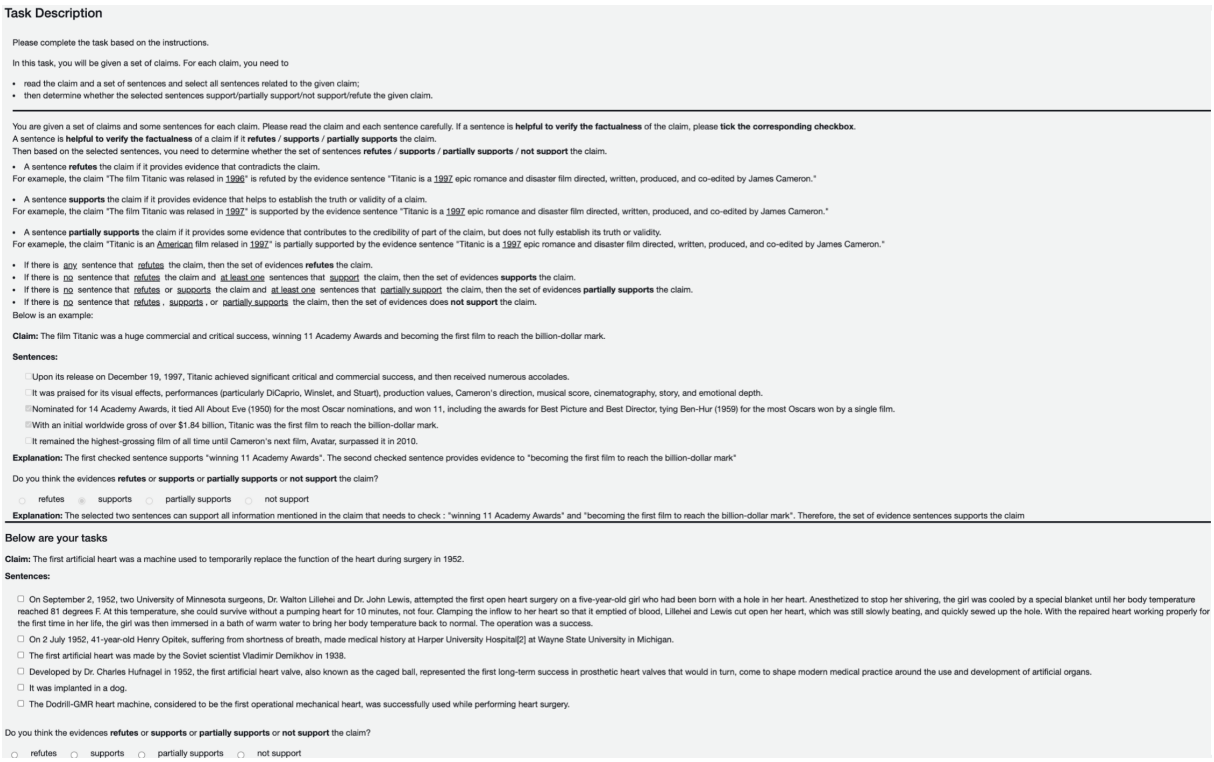

Figure 10: Design of human annotation for evidence retrieval and veracity prediction

User Query: What inspired the invention of the first artificial heart?

LLM Response: The invention of the first artificial heart was inspired by the need to save lives of people with heart failure and to overcome the shortage of donor hearts for transplantation. According to, the first artificial heart was a machine that was used to temporarily replace the function of the heart during surgery in 1952. The first artificial heart that was implanted in a human was the Jarvik-7 in 1982, designed by a team including Willem Johan Kolff, William DeVries and Robert Jarvik. The first patient to receive the Jarvik-7 was Barney Clark, a dentist from Seattle, who survived for 112 days after the implantation.

Fact-checking Annotation:

Claim 1: The invention of the first artificial heart was inspired by the need to save lives of people with heart failure and to overcome the shortage of donor hearts for transplantation.
Search Query: Artificial heart invention inspiration and purpose
Evidence:
The purpose is to develop a temporary machine or pump for a person who has a disease of the heart and their survival without transplant is impossible.
These temporary devices can provide enough time for the patient until a donor heart is available.
.....
Label: Supported

Claim 2: The first artificial heart was a machine that was used to temporarily replace the function of the heart during surgery in 1952.
Search Query: first artificial heart function replacement machine surgery 1952
Evidence:
In 1952, the Model II heart-lung device was released to be used on humans.
Label: Not Supported

Claim 3: The first artificial heart that was implanted in a human was the Jarvik-7 in 1982.
Search Query: first artificial heart implanted in human Jarvik-7 1982
Evidence:
The 61-year-old retired dentist was in an advanced stage of cardiomyopathy, a progressive weakening of the heart muscle, when he became the first recipient of a permanent artificial heart on this day, Dec. 2, in 1982.
.....
Label: Supported

Claim 4: The Jarvik-7 was designed by a team including Willem Johan Kolff, William DeVries, and Robert Jarvik.
Search Query: Jarvik-7 artificial heart design team members
Evidence:
Jarvik completed two years of study, and in 1971 was hired by Willem Johan Kolff, a Dutch-born physician-inventor at the University of Utah,who produced the first dialysis machine, and who was working on other artificial organs, including a heart.
.....
Label: Partially Supported

(next page)

(Continued)

Claim 5: The first patient to receive the Jarvik-7 was Barney Clark, a dentist from Seattle.
Search Query: Jarvik-7 first patient Barney Clark Seattle
Evidence:
On December 2, 1982, Clark became the world2̆019s first recipient of an artificial heart.
The 61-year-old retired dentist was in an advanced stage of cardiomyopathy, a progressive weakening of the heart muscle, when he became the first recipient of a permanent artificial heart on this day, Dec. 2, in 1982.
.....
Label: Supported

Claim 6: Barney Clark survived for 112 days after the implantation of the Jarvik-7.
Search Query: Barney Clark Jarvik-7 implantation survival duration
Evidence:
Barney Clark survived for 112 days after the implantation of the Jarvik-7.
On 1 December 1982, William DeVries implanted the artificial heart into retired dentist Barney Bailey Clark (born 21 January 1921), who survived 112 days with the device, dying on 23 March 1983.
Label: Supported

Figure 11: An example in BINGCHECK. A record contains a user query, original LLM response, and fact-checking annotation. The fact-checking annotation involves claims to verify, search queries, search results, selected evidence, and verdict labels. The search results and the part of selected evidence are omitted due to space limit.

Model	Golden Evidence		FEVER Score	Accuracy	Evidence Retrieval		
	Document	Sentence			F1	Precision	Recall
SELF-CHECKER	✗	✗	51.3	62.7	55.9	64.1	49.5
	✓	✗	64.1	72.8	75.8	90.3	65.4
		✓	-	81.20	-	-	-

Table 8: Ablation results on entire FEVER test set. "Golden Evidence" indicates whether the golden documents/sentences are provided.

Model	Golden Evidence	Claim Split	F1	Accuracy	Evidence Retrieval		
					F1	Precision	Recall
T5-3B (Kamoi et al., 2023)	✗	✓	65.3	77.1	67.4	65.0	81.7
	✓	✓	78.0	84.4	-	-	-
SELF-CHECKER	✗	✓	64.4	68.4	42.1	70.2	30.1
	✗	✗	35.6	35.8	23.5	91.1	13.5
	✓	✓	78.7	78.8	-	-	-
	✓	✗	71.5	47.7	-	-	-

Table 9: Ablation results on WiCE. "Golden Evidence" indicates whether golden sentences are provided. "Claim Split" indicates whether claim decomposition is performed. Note that we compare with the model finetuned on WiCE dataset (Kamoi et al., 2023).

C Ablation Study

Ablation Study Results on FEVER dataset Comparing the first and second rows of Table 8, we observe substantial improvements across all metrics when predicted documents are replaced with golden evidence documents. This improvement suggests the importance of exploring more effective strategies for generating appropriate search queries and improving document retrieval accuracy. Furthermore, the inclusion of golden evidence sentences can further improve the accuracy of veracity prediction by more than 8 points. However, even with golden evidence sentences, the SELF-CHECKER lags behind the SOTA model in label accuracy, indicating the need for further enhancements in the verdict counselor's performance.

Ablation Study Results on Wice Dataset The evaluation results on WiCE dataset is shown in Table 9. The slight improvement in verdict prediction between the first and third rows of the SELF-CHECKER results suggests that the evidence seeker module's efficiency is unlikely to be the primary bottleneck in the SELF-CHECKER's performance. However, comparing the second row of the baseline with the third row of the SELF-CHECKER results highlights that the verdict counselor module's performance is the primary bottleneck in the overall performance of SELF-CHECKER. This finding aligns with the results obtained on the FEVER dataset, indicating the significant potential for enhancing verdict prediction despite LLMs' superior capabilities in various NLP tasks. Consistent with prior findings (Kamoi et al., 2023), we find that decomposing complex claims into simpler sub-claims improves both evidence retrieval and verdict prediction.