Abstract

Fact-checking real-world claims often requires complex, multi-step reasoning due to the absence of direct evidence to support or refute them. However, existing fact-checking systems often lack transparency in their decision-making, making it challenging for users to comprehend their reasoning process. To address this, we propose the Question-guided Multi-hop Fact-Checking (QACHECK) system, which guides the model’s reasoning process by asking a series of questions critical for verifying a claim. QACHECK has five key modules: a claim verifier, a question generator, a question-answering module, a QA validator, and a reasoner. Users can input a claim into QACHECK, which then predicts its veracity and provides a comprehensive report detailing its reasoning process, guided by a sequence of (question, answer) pairs. QACHECK also provides the source of evidence supporting each question, fostering a transparent, explainable, and user-friendly fact-checking process.

1 Introduction

In our age characterized by large amounts of both true and false information, fact-checking is not only crucial for countering misinformation but also plays a vital role in fostering trust in AI systems. However, the process of validating real-world claims is rarely straightforward. Unlike the simplicity of supporting or refuting a claim with a single piece of direct evidence, real-world claims often resemble multi-layered puzzles that require complex and multi-step reasoning to solve (Jiang et al., 2020; Nguyen et al., 2020; Aly and Vlachos, 2022; Chen et al., 2022; Pan et al., 2023).

As an example, to verify the claim “Sunlight can reach the deepest part of the Black Sea.”, it may be challenging to find direct evidence on the web that refutes or supports this claim. Instead, a human fact-checker needs to decompose the claim, gather multiple pieces of evidence, and perform step-by-step reasoning (Pan et al., 2023). This reasoning process can be formulated as question-guided reasoning, where the verification of the claim is guided by asking and answering a series of relevant questions, as shown in Figure 1. In this example, we sequentially raise two questions: “What is the greatest depth of the Black Sea?” and “How far can sunlight penetrate water?”. After independently answering these two questions by gathering relevant information from the Web, we can assert that the initial claim is false with simple reasoning.

While several models (Liu et al., 2020; Zhong et al., 2020; Aly and Vlachos, 2022) have been proposed to facilitate multi-step reasoning in fact-checking, they generally lack transparency in their reasoning processes. These models simply take a claim as input, then output a veracity label without an explicit explanation. Recent attempts, such as Quin+ (Samarinas et al., 2021) and WhatTheWikiFact (Chernyavskiy et al., 2021), have aimed to develop more explainable fact-checking systems, by...
searching and visualizing the supporting evidence for a given claim. However, these systems primarily validate a claim from a single document, and do not provide a detailed, step-by-step visualization of the reasoning process as shown in Figure 1.

We introduce the Question-guided Multi-hop Fact-Checking (QACHECK) system, which addresses the aforementioned issues by generating multi-step explanations via question-guided reasoning. To facilitate an explainable reasoning process, QACHECK manages the reasoning process by guiding the model to self-generate a series of questions vital for claim verification. Our system, as depicted in Figure 2, is composed of five modules: 1) a claim verifier that assesses whether sufficient information has been gathered to verify the claim, 2) a question generator to generate the next relevant question, 3) a question-answering module to answer the raised question, 4) a QA validator to evaluate the usefulness of the generated (Q, A) pair, and 5) a reasoner to output the final veracity label based on all collected contexts.

QACHECK offers enough adaptability, allowing users to customize the design of each module by integrating with different models. For example, we provide three alternative implementations for the QA component: the retriever–reader model, the FLAN-T5 model, and the GPT3-based reciter–reader model. Furthermore, we offer a user-friendly interface for users to fact-check any input claim and visualize its detailed question-guided reasoning process. The screenshot of our user interface is shown in Figure 4. We will discuss the implementation details of the system modules in Section 3 and some evaluation results in Section 4. Finally, we present the details of the user interface in Section 5.

2 Related Work

Fact-Checking Systems. The recent surge in automated fact-checking research aims to mitigate the spread of misinformation. Various fact-checking systems, for example, TANBIH\(^2\) (Zhang et al., 2019), PRTA\(^3\) (Martino et al., 2020), and WHATTHEWIKIFACT\(^4\) (Chernyavskiy et al., 2021) predominantly originating from Wikipedia and claims within political or scientific domains, have facilitated this endeavor. However, the major-

\(^2\)https://www.tanbih.org/about
\(^3\)https://propaganda.qcri.org/
\(^4\)https://www.tanbih.org/whatthewikifact

Figure 2: The architecture of our QACHECK system.

265
Verifier $D$, a Question Generator $Q$, a Question-Answering Model $A$, a Validator $V$, and a Reasoner $R$. We first initialize an empty context $C = \emptyset$. Upon the receipt of a new input claim $c$, the system first utilizes the claim verifier to determine the sufficiency of the existing context to validate the claim, i.e., $D(c, C) \rightarrow \{\text{True}, \text{False}\}$. If the output is False, the question generator learns to generate the next question that is necessary for verifying the claim, i.e., $Q(c, C) \rightarrow q$. The question-answering model is then applied to answer the question and provide the supported evidence, i.e., $A(q) \rightarrow a, e$, where $a$ is the predicted answer, and $e$ is the retrieved evidence that supports the answer. Afterward, the validator is used to validate the usefulness of the newly-generated $(Q, A)$ pair based on the existing context and the claim, i.e., $V(c, (q, a), C) \rightarrow \{\text{True}, \text{False}\}$. If the output is True, the $(q, a)$ pair is added into the context $C$. Otherwise, the question generator is asked to generate another question. We repeat this process of calling $D \rightarrow Q \rightarrow A \rightarrow V$ until the claim verifier returns a True indicating that the current context $C$ contains sufficient information to verify the claim $c$. In this case, the reasoner module is called to utilize the stored relevant context to justify the veracity of the claim and outputs the final label, i.e., $R(c, C) \rightarrow \{\text{Supported}, \text{Refuted}\}$. The subsequent sections provide a comprehensive description of the five key modules in QACHECK.

3.1 Claim Verifier

The claim verifier is a central component of QACHECK, with the specific role of determining if the current context information is sufficient to verify the claim. This module is to ensure that the system can efficiently complete the claim verification process without redundant reasoning. We build the claim verifier based on InstructGPT (Ouyang et al., 2022), utilizing its powerful in-context learning ability. Recent large language models such as InstructGPT (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023) have demonstrated strong few-shot generalization ability via in-context learning, in which the model can efficiently learn a task when prompted with the instruction of the task together with a small number of demonstrations. We take advantage of InstructGPT’s in-context learning ability to implement the claim verifier. We prompt InstructGPT with ten distinct in-context examples as detailed in Appendix A.1, where each example consists of a claim and relevant question–answer pairs. We then prompt the model with the claim, the context, and the following instruction:

\[
\text{Claim} = \text{CLAIM} \\
\text{We already know the following:}
\]

\[
\text{CONTEXT} \\
\text{Can we know whether the claim is true or false now? Yes or no?}
\]

If the response is ‘no’, we proceed to the question generator module. Conversely, if the response is ‘yes’, the process jumps to call the reasoner module.

3.2 Question Generator

The question generator module is called when the initial claim lacks the necessary context for verification. This module aims to generate the next relevant question needed for verifying the claim. Similar to the claim verifier, we also leverage InstructGPT for in-context learning. We use slightly different prompts for generating the initial question and the follow-up questions. The detailed prompts are in Appendix A.2. For the initial question generation, the instruction is:

\[
\text{Claim} = \text{CLAIM} \\
\text{To verify the above claim, we can first ask a simple question:}
\]

For follow-up questions, the instruction is:

\[
\text{Claim} = \text{CLAIM} \\
\text{We already know the following:}
\]

\[
\text{CONTEXT} \\
\text{To verify the claim, what is the next question we need to know the answer to?}
\]

3.3 Question Answering Model

After generating a question, the Question Answering (QA) module retrieves corresponding evidence and provides an answer as the output. The system’s reliability largely depends on the accuracy of the QA module’s responses. Understanding the need for different QA methods in various fact-checking scenarios, we introduce three different implementations for the QA module, as shown in Figure 3.

Retriever–Reader. We first integrate the well-known retriever–reader framework, a prevalent QA paradigm originally introduced by Chen et al. (2017). In this framework, a retriever first retrieves relevant documents from a large evidence
corpus, and then a reader predicts an answer conditioned on the retrieved documents. For the evidence corpus, we use the Wikipedia dump provided by the Knowledge-Intensive Language Tasks (KILT) benchmark (Petroni et al., 2021), in which the Wikipedia articles have been pre-processed and separated into paragraphs. For the retriever, we apply the widely-used sparse retrieval based on BM25 (Robertson and Zaragoza, 2009), implemented with the Pyserini toolkit (Lin et al., 2021). For the reader, we use the RoBERTa-large (Liu et al., 2019) model fine-tuned on the SQuAD dataset (Rajpurkar et al., 2016), using the implementation from PrimeQA\(^5\) (Sil et al., 2023).

**FLAN-T5.** While effective, the retriever–reader framework is constrained by its reliance on the evidence corpus. In scenarios where a user’s claim is outside the scope of Wikipedia, the system might fail to produce a credible response. To enhance flexibility, we also incorporate the FLAN-T5 model (Chung et al., 2022), a Seq2Seq model pre-trained on more than 1.8K tasks with instruction tuning. It directly takes the question as input and then generates the answer and the evidence, based on the model’s parametric knowledge.

**GPT Reciter–Reader.** Recent studies (Sun et al., 2023; Yu et al., 2023) have demonstrated the great potential of the GPT series, such as InstructGPT (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023), to function as robust knowledge repositories. The knowledge can be retrieved by properly prompting the model. Drawing from this insight, we introduce the GPT Reciter–Reader approach. Given a question, we prompt the InstructGPT to “recite” the knowledge stored within it, and InstructGPT responds with relevant evidence. The evidence is then fed into a reader model to produce the corresponding answer. While this method, like FLAN-T5, does not rely on a specific corpus, it stands out by using InstructGPT. This offers a more dependable parametric knowledge base than FLAN-T5.

The above three methods provide a flexible and robust QA module, allowing for switching between the methods as required, depending on the claim being verified and the available contextual information. In the following, we use GPT Reciter–Reader as the default implementation for our QA module.

### 3.4 QA Validator

The validator module ensures the usefulness of the newly-generated QA pairs. For a QA pair to be valid, it must satisfy two criteria: 1) it brings additional information to the current context \(C\), and 2) it is useful for verifying the original claim. We again implement the validator by prompting InstructGPT with a suite of ten demonstrations shown in Appendix A.3. The instruction is as follows:

\[
\text{Claim} = \text{CLAIM} \\
\text{We already know the following:} \text{CONTEXT} \\
\text{Now we further know:} \text{NEW QA PAIR} \\
\text{Does the QA pair have additional knowledge useful for verifying the claim?}
\]

The validator acts as a safeguard against the system producing redundant or irrelevant QA pairs. Upon validation of a QA pair, it is added to the current context \(C\). Subsequently, the system initiates another cycle of calling the claim verifier, question generator, question answering, and validation.
2. Submit to fact-check the input claim

Reasoning depth: 0
Generated Question: In which year did Lars Onsager win the Nobel prize?

The Nobel Prize in Chemistry 1968 was awarded to Lars Onsager for the discovery of the reciprocal relations bearing his name, which are fundamental for the thermodynamics of irreversible processes.

Predicted Answer: 1968

3. Visualize the question-answering guided reasoning process

Lars Onsager won the Nobel prize when he was 30 years old.

Lars Onsager won the Nobel prize in 1968. Lars Onsager was born in 1903. He was 65 when he won the Nobel prize. Therefore, the final answer is: False.

4. The final prediction result with rationale

Lars Onsager won the Nobel prize in 1968. Lars Onsager was born in 1903. He was 65 when he won the Nobel prize. Therefore, the final prediction is: False.

3.5 Reasoner

The reasoner is called when the claim verifier determines that the context $C$ is sufficient to verify the claim or the system hits the maximum allowed iterations, set to 5 by default. The reasoner is a special question-answering model which takes the context $C$ and the claim $c$ as inputs and then answers the question "Is the claim true or false?". The model is also requested to output the rationale with the prediction. We provide two different implementations for the reasoner: 1) the end-to-end QA model based on FLAN-T5, and 2) the InstructGPT model with the prompts given in Appendix A.4.

4 Performance Evaluation

To evaluate the performance of our QACHECK, we use two fact-checking datasets that contain complex claims requiring multi-step reasoning: HOVER (Jiang et al., 2020) and FEVEROUS (Aly et al., 2021), following the same experimental set-
Table 1: Evaluation of F1 scores for different models. The bold text shows the best results for each setting.

<table>
<thead>
<tr>
<th>Model</th>
<th>HOVER 2-hop</th>
<th>HOVER 3-hop</th>
<th>HOVER 4-hop</th>
<th>FEVEROUS 2-hop</th>
<th>FEVEROUS 3-hop</th>
<th>FEVEROUS 4-hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstructGPT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Direct</td>
<td>56.51</td>
<td>51.75</td>
<td>49.68</td>
<td>60.13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- CoT</td>
<td><strong>57.20</strong></td>
<td>53.66</td>
<td>51.83</td>
<td><strong>61.05</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Codex</td>
<td>55.57</td>
<td>53.42</td>
<td>45.59</td>
<td>57.85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLAN-T5</td>
<td>48.27</td>
<td>52.11</td>
<td>51.13</td>
<td>55.16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProgramFC</td>
<td>54.27</td>
<td>54.18</td>
<td><strong>52.88</strong></td>
<td>59.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QACheck</td>
<td><strong>55.67</strong></td>
<td><strong>54.67</strong></td>
<td>52.35</td>
<td>59.47</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Upon selecting or inputting a claim, the user can start the fact-checking process by clicking the “Submit” button. The bottom half of Figure 4 shows a snapshot of QACHECK’s output for the input claim “Lars Onsager won the Nobel prize when he was 30 years old”. The system visualizes the detailed question-guided reasoning process. For each reasoning step, the system shows the index of the reasoning step, the generated question, and the predicted answer to the question. The retrieved evidence to support the answer is shown on the right for each step. The system then shows the final veracity prediction for the original claim accompanied by a comprehensive rationale in the “Prediction with rationale” section. This step-by-step illustration not only enhances the understanding of our system’s fact-checking process but also offers transparency to its functioning.

QACheck also allows users to change the underlying question–answering model. As shown at the top of Figure 4, users can select between the three different QA models introduced in Section 3.3, depending on their specific requirements or preferences. Our demo system will be open-sourced under the Apache-2.0 license.

6 Conclusion and Future Works

This paper presents the QACheck system, a novel approach designed for verifying real-world complex claims. QACheck conducts the reasoning process with the guidance of asking and answering a series of questions and answers. Specifically, QACheck iteratively generates contextually relevant questions, retrieves and validates answers, judges the sufficiency of the context information, and finally, reasons out the claim’s truth value based on the accumulated knowledge. QACheck leverages a wide range of techniques, such as in-context learning, document retrieval, and question-answering, to ensure a precise, transparent, explainable, and user-friendly fact-checking process.

In the future, we plan to enhance QACheck 1) by integrating additional knowledge bases to further improve the breadth and depth of information accessible to the system (Feng et al., 2023; Kim et al., 2023), and 2) by incorporating a multi-modal interface to support image (Chakraborty et al., 2023), table (Chen et al., 2020; Lu et al., 2023), and chart-based fact-checking (Akhtar et al., 2023), which can broaden the system’s utility in processing and analyzing different forms of data.

5 User Interface

We create a demo system based on Flask6 for verifying open-domain claims with QACheck, as shown in Figure 4. The QACheck demo is designed to be intuitive and user-friendly, enabling users to input any claim or select from a list of pre-defined claims (top half of Figure 4).

---

6https://flask.palletsprojects.com/en/2.3.x/
Limitations

We identify two main limitations of QACHECK. First, several modules of our QACHECK currently utilize external API-based large language models, such as InstructGPT. This reliance on external APIs tends to prolong the response time of our system. As a remedy, we are considering the integration of open-source, locally-run large language models like LLaMA (Touvron et al., 2023). Secondly, the current scope of our QACHECK is confined to evaluating True/False claims. Recognizing the significance of also addressing Not Enough Information claims, we plan to devise strategies to incorporate these in upcoming versions of the system.

Ethics Statement

The use of large language models requires a significant amount of energy for computation for training, which contributes to global warming. Our work performs few-shot in-context learning instead of training models from scratch, so the energy footprint of our work is less. The large language model (InstructGPT) whose API we use for inference consumes significant energy.

Acknowledgement

This project is supported by the Ministry of Education, Singapore, under its MOE AcRF TIER 3 Grant (MOE-MOET32022-0001). The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore.

References


A Prompts

A.1 Prompts for Claim Verifier

Claim = Superdrag and Collective Soul are both rock bands.
We already know the following:
Question 1 = Is Superdrag a rock band?
Answer 1 = Yes
Can we know whether the claim is true or false now? Yes or no?
Prediction = No, we cannot know.

Claim = Superdrag and Collective Soul are both rock bands.
We already know the following:
Question 1 = Is Superdrag a rock band?
Answer 1 = Yes
Question 2 = Is Collective Soul a rock band?
Answer 2 = Yes
Can we know whether the claim is true or false now? Yes or no?
Prediction = Yes, we can know.

A.2 Prompts for Question Generation

Prompts for the initial question generation
Claim = Superdrag and Collective Soul are both rock bands.
To verify the above claim, we can first ask a simple question:
Question = Is Superdrag a rock band?

Prompts for the follow-up question generation
Claim = Superdrag and Collective Soul are both rock bands.
We already know the following:
Question 1 = Is Superdrag a rock band?
Answer 1 = Yes
To verify the claim, what is the next question we need to know the answer to?
Question 2 = Is Collective Soul a rock band?

A.3 Prompts for Validator

Claim = Superdrag and Collective Soul are both rock bands.
We already know the following:
Question = Is Superdrag a rock band?
Answer = Yes
Now we further know:

Question = Is Collective Soul a rock band?
Answer = Yes
Does the QA pair have additional knowledge useful for verifying the claim?
The answer: Yes

A.4 Prompts for Reasoner

Contexts:
Q1: When Lars Onsager won the Nobel Prize?
A1: 1968
Q2: When was Lars Onsager born?
A2: 1903
Claim = Lars Onsager won the Nobel Prize when he was 30 years old.

Is this claim true or false?
Answer: Lars Onsager won the Nobel Prize in 1968.
Lars Onsager was born in 1903.
Therefore, the final answer is: False.

Contexts:
[[CONTEXTS]]
Claim = [[CLAIM]]
Is this claim true or false?
Answer:
Therefore, the final answer is