Enhancing Fact-Checking in Large Language Models: Cost-Effective Claim Verification through First-Order Logic Reformulation

by

Sara Asghari
B.Sc. (Computer Software Engineering), AmirKabir Univeristy of Technology, 2022

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Enhancing Fact-Checking in Large Language Models: Cost-Effective Claim Verification through First-Order Logic Reformulation

by

Sara Asghari

B.Sc. (Computer Software Engineering), Amirkabir University of Technology, 2022

Supervisory Committee

Dr. A. Thomo, Supervisor

(Department of Computer Science)

Dr. V. Srinivasan, Co-supervisor

(Department of Computer Science)

ABSTRACT

In the realm of Large Language Models (LLMs), the ability to accurately perform Fact Checking (FC) tasks, which involves verifying complex claims against challenging evidence from multiple sources, remains a crucial yet under-explored area. Our study presents a comprehensive benchmarking of various LLMs, including GPT-4, on this critical task. We utilize a modern, challenging dataset designed explicitly for fact-checking, HOVER, which comprises thousands of evidence-claim pairs covering diverse aspects of life, history, and entertainment. This dataset differs from common datasets that evaluate the reading comprehension capabilities of LLMs, which are primarily composed of sets of question-and-answer pairs.

Our findings demonstrate that GPT-4 not only decisively surpasses the current state-of-the-art (SOTA) models in FC tasks but also shows that other, open-source, LLMs (e.g. Mixtral and Llama-3) exhibit close-to-SOTA performance out-of-the-box. This implies that simply presenting these models with the evidence text and claim allows them to infer the claim's veracity effectively. We contrast this with the existing SOTA methods, which involve complex, multi-step solutions, including the use of multiple LLMs to verify claims – a process that necessitates continuous updates and local execution, making it less accessible for regular users.

Furthermore, we explore the impact of claim formulation on the FC task's effectiveness. By converting complex claims into first-order logic (FOL) and then back into natural language, we observe improved performance in some LLMs, particularly with more challenging dataset subsets. This method, although utilizing GPT-4 for the FOL breakdown, serves as a practical guideline for users: more formally structured claims yield more reliable responses.

Keywords: Fact Checking, Claim, Evidence, First Order Logic, Large Language Models, Prompt Engineering

Table of Contents

List of Tables

ACKNOWLEDGEMENTS

I would like to extend my deepest gratitude to:

Dr. Thomo, Words cannot fully convey my appreciation for the opportunity you've provided me to pursue this degree. Your guidance and encouragement have been invaluable in shaping my academic path, serving as a constant source of motivation, especially during the most challenging moments.

Dr. Srinivasan, I am sincerely grateful for your generous sharing of knowledge and expertise. Your mentorship has profoundly shaped my understanding of the field. The patience and wisdom you've offered have been key to my growth both academically and personally.

My family, I cannot thank you enough for your unwavering support throughout this journey. Your belief in me has been the foundation of my perseverance. Your sacrifices and encouragement have made this achievement possible, and it belongs as much to you as it does to me.

Chapter 1

Introduction

In recent years, large language models (LLMs) have revolutionized the field of natural language processing (NLP), achieving remarkable milestones in various tasks such as language translation, text generation, and sentiment analysis. However, the task of fact-checking (FC) — the ability to verify the accuracy of complex claims against diverse and often challenging evidence — remains a critical yet under-explored domain. As misinformation proliferates across digital platforms, the need for robust fact-checking mechanisms becomes increasingly vital to uphold the integrity of information. Fact-checking is inherently a complex task that requires sophisticated reasoning and access to a vast array of knowledge sources.

Unlike traditional reading comprehension tasks, which primarily involve answering straightforward questions based on given texts, fact-checking demands a nuanced analysis of claims and evidence drawn from multiple, often disparate sources. This complexity underscores the importance of developing and benchmarking LLMs capable of performing FC tasks with high accuracy and reliability.

Our study — called **F**irst-**O**rder-**L**ogic-**D**ecomposition and **Co**st-Effective Claim **Ve**rification (FOLDCoVe) — addresses this imperative by conducting a comprehen-

sive benchmarking of various LLMs, including GPT-4, specifically for fact-checking tasks. Our findings reveal that GPT-4 not only surpasses the current state-of-the-art (SOTA) models in FC tasks but also demonstrates the near-SOTA performance of other open-source LLMs, such as Mixtral and Llama-3, even when used out-of-the-box. This suggests that these models can effectively infer the veracity of claims simply by being presented with the relevant evidence text. In contrast, existing SOTA methods often involve intricate, multi-step processes, including the use of multiple LLMs for claim verification, which require continuous updates and local execution. Such complexity renders these methods less accessible to regular users.

In addition to benchmarking performance, our study explores the impact of claim formulation on the effectiveness of fact-checking. By employing a detailed process of information decomposition, we convert complex claims into first-order logic (FOL) representations. This involves three sequential steps: interpretation and predicate definition, FOL translation, and anglicization back into natural language. By breaking down claims into their atomic propositions, we observe improved performance in certain LLMs, particularly with more challenging subsets of the dataset. This approach, which utilizes GPT-4 for the FOL breakdown, provides practical guidelines for users: formally structured claims tend to yield more reliable responses from LLMs.

To enhance efficiency, we employ a cost-effective weaker language model (WLM) to verify the anglicized claim derived from the FOL representation. This strategy reduces reliance on computationally expensive models, as the WLM effectively assesses the restructured claim in a single step without verifying each sub-claim separately.

Furthermore, our study reveals that LLMs can effectively perform fact-checking by leveraging their internal knowledge bases, reducing the need for external evidence. Particularly, models like GPT-4 demonstrate the ability to utilize their pre-existing knowledge—acquired during extensive pre-training on sources such as Wikipedia —

to verify claims accurately. This inherent capability means that supplying external evidence is often unnecessary, simplifying the fact-checking process and highlighting the models' advanced internal reasoning abilities.

1.1 Contributions

The main contributions of this study are summarized as follows:

1. **Benchmarking LLMs for Fact-Checking:** We benchmark LLMs like GPT-4, Mixtral, and Llama-3, showing GPT-4 surpasses SOTA models, while open-source LLMs achieve near-SOTA performance, verifying claims effectively with minimal configuration.

2. **Impact of Claim Formulation:** We show that converting claims into FOL and back into natural language boosts the performance of LLMs on complex datasets, demonstrating that structured claims lead to more accurate and reliable fact-checking.

3. **Cost-Efficient Fact-Checking with WLMs:** We emphasize cost-efficiency in fact-checking by utilizing WLMs for verification. By minimizing reliance on computationally expensive models, our approach ensures scalable fact-checking, reducing resource consumption while maintaining accuracy.

4. **Leveraging Internal Knowledge for Fact-Checking:** We reveal that LLMs, such as GPT-4, can use pre-trained internal knowledge from sources like Wikipedia to verify claims without external evidence, simplifying the fact-checking process.

1.2 Organization

In summary, our study underscores the significant advancements in LLM capabilities for fact-checking tasks, highlighting GPT-4's superior performance and the promising potential of open-source models. It also emphasizes the importance of claim formulation in enhancing the accuracy of LLM responses, providing valuable insights for future research and practical applications in the realm of fact-checking.

This thesis is organized as follows:

- **Chapter 2, Preliminaries,** presents the key notations and definitions of FOL used throughout this work. It also offers a thorough background on language models and methods for their optimization, along with an overview of the current SOTA approaches for handling complex fact-checking tasks.

- **Chapter 3, Methodologies,** details two SOTA methods alongside our proposed approach, and provides an in-depth comparison, focusing on the theoretical foundations and methodologies behind each approach.

- **Chapter 4, Experiments,** presents our experimental setup and results, followed by a comparative analysis of the performance of the methods discussed in Chapter 3.

- **Chapter 5, Conclusions and Future Work,** concludes the thesis and offers insights on potential improvements in fact-checking approaches.

Chapter 2

Preliminaries

In this chapter, we give the foundational definitions and concepts that underpin the discussions and analyses in this thesis. We begin by introducing the essential terminology, including formal systems such as first-order logic (FOL), which allows us to reason about relationships between objects and their properties. This is followed by a description of quantifiers, logical connectives, and other key constructs. Additionally, we explore background concepts such as large language models (LLMs), retrieval-augmented generation (RAG), and prompt engineering, which play a critical role in various natural language processing tasks. The following sections provide the necessary tools for understanding the methodologies and techniques discussed in subsequent chapters.

2.1 Definitions

In this section, we begin by introducing some terminology and definitions.

Definition 1. First-order logic (FOL) *[7] is a formal system used to represent knowledge and reason about relationships between objects and their properties. It*

builds upon propositional logic by introducing quantifiers and logical connectives as logical symbols, and functions and predicates (relations) as non-logical symbols.

Definition 2. Quantifiers *allow us to reason about all objects or specific objects in a domain.*

- $\forall$ **(Universal Quantifier)***: Represents the phrase "for all" or "for every". For example,* $\forall x P(x)$ *means "For all* x*,* $P(x)$ *is true", where* $P(x)$ *is a property of* x*.*

- $\exists$ **(Existential Quantifier)***: Represents the phrase "there exists" or "there is at least one". For example,* $\exists x P(x)$ *means "There exists an* x *such that* $P(x)$ *is true".*

Definition 3. Logical connectives *enable us to combine statements and express complex relationships.*

- $\wedge$ **(Conjunction)***: Represents the word "and". For example,* $P(x) \wedge Q(x)$ *means "*$P(x)$ *is true and* $Q(x)$ *is true".*

- $\vee$ **(Disjunction)***: Represents the word "or". For example,* $P(x) \vee Q(x)$ *means "*$P(x)$ *is true or* $Q(x)$ *is true".*

- $\implies$ **(Implication)***: Represents the phrase "if...then". For example,* $P(x) \implies Q(x)$ *means "If* $P(x)$ *is true, then* $Q(x)$ *is true".*

- $\iff$ **(Biconditional)***: Represents the phrase "if and only if". For example,* $P(x) \iff Q(x)$ *means "P(x) is true if and only if* $Q(x)$ *is true".*

- $\neg$ **(Negation)***: Represents the word "not". For example,* $\neg P(x)$ *means "P(x) is not true".*

Definition 4. Functions *establish a mapping between objects, by taking a set of objects as input and returning a unique single object as output. For example, the function "MotherOf(x)" where x is a person, returns x's mother. Functions can have varying numbers of arguments, also known as arity or valence. A function with zero arguments is called a constant, as it consistently returns the same object regardless of input.*

Definition 5. Predicates (Relations) *take a set of objects as input and return a truth value (true or false) depending on whether the input objects satisfy the specified property or relationship. For instance, the predicate "IsEmployed(x)" assesses whether a person x is employed, while the predicate "IsTallerThan(x, y)" determines if person x is taller than person y.*

Example: Illustrating the Breakdown of an English Paragraph Into First-Order Logic

Claim: Elon Musk, the CEO of Tesla, known for criticizing hydrogen as "dumb" for energy storage, surprisingly announced Tesla's shift to hydrogen power by 2026. This decision, a response to competition from BYD, aims to diversify Tesla's product offerings. The first hydrogen-powered car, Model H, utilizing fuel cells, is expected to debut in 2026.

First-Order Logic Representation:

- **Predicates:**

  CEO(x,y): x is the CEO of company y

  Criticized(x,y,z): x criticized y as z

  Announced(x,y,z): x announced event y at time z

  ShiftTo(x,y): x shifts to technology y

  ResponseTo(x,y): x is a response to y

Competition(x,y): x is a competitor of y

Aim(x,y): x aims to achieve y

DiversifyProductOfferings(x): x diversifies its product offerings

Uses(x,y): x uses technology y

FuelCells(x): x is a fuel cell

ExpectedDebut(x,y): x is expected to debut at time y

- **Formula:**

  CEO("Elon Musk", "Tesla") $\wedge$ Criticized("Elon Musk", "Hydrogen Power", "Dumb") $\wedge$ Announced("Elon Musk", ShiftTo("Tesla", "Hydrogen Power"), 2026) $\wedge$ ResponseTo(ShiftTo("Tesla", "Hydrogen Power"), Competition("BYD", "Tesla")) $\wedge$ Aim(ShiftTo("Tesla", "Hydrogen Power"), DiversifyProductOfferings("Tesla")) $\wedge$ Uses("Model H", "Fuel Cells") $\wedge$ ExpectedDebut("Model H", 2026)

2.2 Background

2.2.1 Language Models

Language Models, either statistical or neural network-based, are sophisticated software systems designed to generate and predict human-like text by learning patterns, structures, and semantics from vast natural language data using self-supervised learning. Their complexity ranges from simple n-gram models, which predict words based on preceding sequences, to advanced transformer-based models like GPT, capable of processing large text volumes and producing coherent, contextually relevant responses. The sophistication and performance of these models are often measured by the number of parameters they possess, with a larger number generally indicating greater capabilities. Primarily built using deep learning techniques, these models

predict word sequence probabilities and are crucial for various natural language processing tasks such as machine translation, text summarization, question-answering, and conversational agents. Through techniques such as fine-tuning and prompt engineering, these models can be adapted for specific tasks [16]. Advanced models like GPT-4 utilize transformer architectures to understand complex dependencies and contextual information, generating coherent and contextually relevant text. These models are essential for artificial intelligence applications requiring human-like language understanding and generation capabilities. In the following, we describe the language models we utilize in this thesis.

GPT-3.5-Turbo

GPT-3.5-Turbo is an advanced iteration of the Generative Pre-trained Transformer (GPT) series developed by OpenAI, released on November 28, 2022. This autoregressive language model represents a significant enhancement over its predecessors in terms of computational efficiency and language understanding capabilities. Built on the transformer architecture and trained on a massive dataset of text and code, it excels in various natural language processing tasks, including text completion, translation, summarization, conversation, and creative content generation. The "Turbo" designation indicates optimizations for faster inference and reduced latency, making it ideal for real-time applications.

Version 1106 of GPT-3.5-Turbo introduces significant improvements, including enhanced instruction following, JSON mode, parallel function calling capabilities, and log probabilities for analyzing decision-making processes. These enhancements result in a 38% improvement in format following tasks like generating JSON, XML, and YAML, as demonstrated by OpenAI's internal evaluations. Additionally, this version boasts an impressive 16.4K token input context window and a maximum

output of 16.4K tokens, enabling substantial context retention and the generation of comprehensive responses.

It's important to note that GPT-3.5-Turbo's knowledge cutoff is September 2021, meaning it doesn't have information on events or developments that occurred after that date. GPT-3.5-Turbo demonstrates improved coherence, context retention, and complex query handling, contributing to advancements in AI-driven communication. Its ability to generate human-like text and provide informative answers makes it a powerful tool for artificial intelligence-driven interaction across various domains.

GPT-4-Turbo

OpenAI introduced the initial version of GPT-4 on March 14, 2023, and followed it with the more advanced GPT-4 Turbo on November 6, 2023. This next-generation model boasts a remarkable 128K token context window, accommodating over 300 pages of text within a single prompt, though the maximum output is limited to 4,096 tokens. Its knowledge base extends to April 2023, providing relevant information for various tasks.

GPT-4 Turbo is significantly larger, with 170 trillion parameters compared to the 175 billion parameters of GPT-3.5. This size difference demonstrates the amplified capabilities in performance and accuracy, particularly in handling complex language models and natural language processing tasks. GPT-4 exhibits human-level performance on various professional and academic benchmarks. For instance, it scored in the top 10% of test-takers on the Uniform Bar Exam, demonstrating its impressive capabilities in the legal domain, whereas GPT-3.5 scored in the bottom 10%.

Moreover, GPT-4 Turbo supports multimodal inputs, including text and images, in contrast to GPT-3.5, which processes only text inputs. One of GPT-4 Turbo's notable features is function calling, enabling users to describe their app functions or

external APIs, prompting the model to generate JSON objects with arguments for invoking those functions. This significantly enhances the model's ability to interact with external systems and perform complex tasks. OpenAI has introduced the response_format API parameter to ensure the model consistently produces syntactically accurate JSON objects, and the seed API parameter allows for reproducible outputs, granting users more control over the model's behavior and ensuring consistent results across multiple interactions.

Despite its advancements, GPT-4 Turbo is considerably more expensive than its predecessor, GPT-3.5-Turbo, with input tokens costing 10 times more and output tokens 15 times more. Nevertheless, its enhanced capabilities and expanded context window make it a valuable asset for applications requiring advanced language understanding and generation in various domains [1, 13].

Llama-3-70B-Chat

The Meta Llama-3-70B model, part of the Meta Llama-3 family of open-source LLMs, represents a significant advancement in the field of natural language processing. Released in April 2024, the Meta Llama-3 family includes both pre-trained (base) and instruction-tuned generative text models available in 8B and 70B sizes.

The instruction-tuned versions are specifically optimized for dialogue and assistant-like chat applications, utilizing supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF) to ensure alignment with human preferences for helpfulness and safety. This fine-tuning process incorporates publicly available instruction datasets along with over 10 million human-annotated examples, enhancing the models' capability to generate contextually appropriate and user-friendly outputs. These models have shown superior performance compared to many open-source chat models on industry benchmarks.

The Llama-3 models exclusively take text as input and are capable of generating both text and code, making them versatile for a range of natural language generation tasks.

Llama-3 is intended for commercial and research use in English. However, developers can fine-tune Llama-3 models for languages beyond English.

A notable feature of the Llama-3-70B model is its pretraining data, which includes over 15 trillion tokens from publicly available sources — a dataset seven times larger than that used for Llama-2 and comprising four times more code. This pretraining data has a cutoff of December 2023.

The Llama-3 models benefit from a context length of 8,192 tokens, which is achieved by training on sequences of the same length, effectively doubling the capacity of their predecessor, Llama-2 [2].

Mixtral-8x7B-Instruct

The Mixtral-8x7B model, developed by Mistral AI and released in December 2023, represents a significant advancement in generative sparse Mixture of Experts (SMOE) technology. As a pre-trained model, it leverages a mixture of eight experts (MoE) composed of 7B models, selecting two experts during inference to optimize performance. This design allows Mixtral-8x7B to efficiently manage 46.7 billion total parameters, while only utilizing 12.9 billion parameters per token, enabling it to process inputs and generate outputs with the speed and cost efficiency of a 12.9 billion parameter model.

The model can handle a context window of 32,000 tokens and effectively retrieve information, regardless of the sequence length or the position of the information within the sequence. Additionally, it can generate a maximum of 4,096 output tokens. Its multilingual training, drawn from diverse data on the open web, allows it to

proficiently handle English, French, Italian, German, and Spanish, making it versatile across various languages.

Mixtral-8x7B-instruct has been fine-tuned through supervised learning and Direct Preference Optimization (DPO) to excel in following instructions meticulously.

In terms of performance, Mixtral-8x7B stands out remarkably, surpassing Llama-2-70B on benchmarks related to mathematics, code generation, and multilingual understanding tasks, while delivering six times faster inference. This highlights Mixtral's efficiency in balancing quality and inference budget compared to the Llama-2 models. It also competes robustly against GPT-3.5, often matching or exceeding its performance on standard benchmarks, and is the first open-weight model to surpass GPT-3.5's performance. Furthermore, Mixtral-8x7B-instruct notably outperforms GPT3.5-Turbo, Claude-2.1, Gemini Pro, and Llama-2-70B-Chat models on human benchmarks. Additionally, a 22B size model has been released as well, offering further options for different performance and resource needs.[10]

2.2.2 Prompt Engineering

Prompt Engineering is the practice of developing and optimizing prompts — text-based inputs provided to large language models (LLMs) to guide their outputs. A prompt serves as the foundation for communication with an LLM, consisting of an instruction and potentially including input data, context, and some indication of the desired output. The primary purpose of a prompt is to equip the language model with the necessary instructions and context to fulfill a specific task effectively. Within the field of LLMs, prompt engineering is an essential discipline focused on crafting prompts in a way that maximizes the accuracy and relevance of the generated outputs, thereby harnessing the full potential of LLMs. This process is not merely about asking questions; it requires a strategic approach to structuring and combining prompt

patterns to address a variety of challenges encountered in output generation and interaction. By providing reusable solutions to common problems, prompt engineering establishes a framework for documenting effective strategies, enabling users to consistently achieve desired results. In essence, prompt engineering is pivotal in controlling and optimizing the outputs of LLMs, making it a critical tool for anyone looking to leverage the capabilities of these advanced models effectively [16].

Zero-Shot Learning

Zero-Shot Learning is an approach in machine learning where language models are conditioned on specific instructions that describe the task they need to perform. This technique, often referred to as zero-shot prompting, involves the use of carefully constructed prompts that guide the model to complete tasks it hasn't been explicitly trained for. Essentially, it is a form of prompt engineering, where templates are designed to extract the desired behavior from the language model. A noteworthy advancement within this framework is the Zero-Shot Chain of Thought (Zero-Shot-CoT) reasoning [12]. This task-agnostic method enhances the model's reasoning capabilities by adding a single fixed phrase, "Let's think step by step", to the prompt. This phrase encourages the model to engage in a step-by-step thought process before arriving at an answer, thereby improving the quality and accuracy of its responses. By leveraging these techniques, zero-shot learning enables models to generalize across various tasks, showcasing their versatility and adaptability.

Few-Shot Learning

Few-Shot Learning is a technique in machine learning that allows language models to perform various tasks by conditioning them on a small set of examples, referred to as few-shot prompts. These prompts contain explicit examples of the task at hand, en-

abling the model to learn and adapt effectively even with limited data. This method is a form of in-context learning, where the model is guided by demonstrations included in the prompt, improving its ability to generate accurate responses for subsequent examples. One notable instance of few-shot prompting is Chain of Thought (CoT) prompting. This technique modifies the answers in few-shot examples to include step-by-step reasoning, rather than just providing standard question-and-answer pairs. By feeding the model with these detailed, interconnected prompts, CoT prompting helps the language model to produce responses that follow a logical and coherent reasoning path. This breakdown of complex reasoning tasks into simpler steps enhances the model's ability to handle intricate problems, making few-shot learning, and specifically CoT prompting, a powerful approach in the advancement of large language models.

2.2.3 Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is a framework that addresses limitations in LLMs, such as hallucination [6] and knowledge staleness. While LLMs are powerful in processing vast datasets [19], they struggle to maintain up-to-date and accurate information. RAG overcomes this by integrating LLMs with external knowledge bases, providing real-time access to factual data [8].

RAG combines the model's internal knowledge with external data sources, resulting in responses that are both accurate and traceable to their origins [21]. There are three main paradigms: Naive RAG, Advanced RAG, and Modular RAG. Naive RAG involves simple retrieval and generation, while Advanced RAG introduces strategies like reranking and context compression to improve retrieval quality [9]. Modular RAG further refines this by adding specialized search modules and optimizing retrieval through fine-tuning [15].

Different retrieval techniques enhance RAG's flexibility, such as iterative retrieval, which refines searches based on generated text, and adaptive retrieval, allowing the model to decide when to retrieve and generate [3].

Compared to fine-tuning, which statically adjusts model behavior, RAG dynamically incorporates new information without retraining, blending parametric and non-parametric memory [14]. Challenges remain, particularly concerning the increasing context length of LLMs and mitigating retrieval errors due to noisy or contradictory data [8].

Example: Phases of RAG in Practice

Consider the question, "Who is the current CEO of Tesla?" Using RAG, the process unfolds in three key phases:

1. **Retrieval Phase:** The model sends a query to an external knowledge base, such as a recent news source. The knowledge base searches its indexed documents and retrieves relevant information about Tesla's leadership.

2. **Augmentation Phase:** The retrieved documents (e.g., an article mentioning Elon Musk as the CEO in 2024) are then passed back to the model as part of an augmented prompt. The model incorporates this up-to-date information into its reasoning process to generate a more accurate and contextually relevant response.

3. **Generation Phase:** Finally, the model generates a response based on the augmented data: "Elon Musk is the CEO of Tesla as of 2024." This phase ensures that the answer is both coherent and based on current information, overcoming the limitations of outdated internal knowledge.

This example shows how RAG enhances the model's output by integrating internal reasoning with real-time data, ensuring accuracy and relevance.

2.3 Problem Statement and Approaches

In real-world fact-checking, verifying the accuracy of a claim often requires gathering multiple sources of evidence and applying complex, multi-step reasoning. It is rarely possible to determine the truth of a claim based on information from a single source. Instead, the process involves identifying relevant pieces of evidence and making a judgment about the claim's veracity based on them. For example, consider the claim, "Elon Musk and the CEO of Facebook both left Harvard University without completing their degrees." Finding a single source that confirms or refutes this claim directly can be difficult. A fact-checker must break it down, retrieve information about both individuals separately, and then combine this evidence to reach a conclusion. This highlights the need for an approach that scales in effectiveness as the complexity of reasoning increases.

In fact-checking tasks, different scenarios of evidence availability can significantly impact the approach taken to verify a claim. In the **Gold-Evidence** setting, the necessary documents that either support or refute the claim are directly provided alongside the claim, allowing the model to make a veracity prediction based on this supplied evidence. In the **Open-Book** setting, the model is responsible for retrieving relevant evidence from a broader corpus before using it to evaluate the truth of the claim. Finally, in the **Closed-Book** setting, the model relies entirely on the knowledge it has stored during its training, without access to external sources, using only its stored information to make a prediction about the claim's accuracy.

Two notable approaches to address the complexity of fact-checking include **Pro-**

gramFC and **FOLK**. ProgramFC [18] decomposes complex claims into smaller sub-tasks, each solved using specific functions. It uses LLMs to generate a reasoning program, which guides the process by assigning sub-tasks like answering questions or verifying simpler claims to appropriate handlers. Similarly, FOLK [20] translates a claim into a FOL clause made up of predicates, each representing a sub-claim to be verified. It grounds the verification process in real-world knowledge by generating questions and retrieving answers from reliable sources like Google or Wikipedia. FOLK then uses LLMs to evaluate the FOL predicates and make a veracity prediction.

Chapter 3

Methodologies

This chapter delves into the methodologies that underpin automated fact-checking, drawing inspiration from recent advancements in natural language processing (NLP) and machine learning. The focus is on three innovative approaches: ProgramFC [18], FOLK [20], and our proposed method, FOLDCoVe, which leverage the in-context learning power of large language models (LLMs) to enhance the accuracy, explainability, and data efficiency of fact-checking systems.

3.1 Program-Guided Fact-Checking (ProgramFC)

ProgramFC introduces a novel paradigm for fact-checking complex claims that often require multi-step reasoning and evidence from multiple sources. The key innovation of ProgramFC lies in its two-stage approach:

3.1.1 Program Generation

In this stage, an LLM is employed to generate a Python-like reasoning program for the input claim. This program is essentially a sequence of instructions, akin to a computer program, that outlines the steps needed to verify the claim. These steps can involve

answering questions, verifying simpler sub-claims, or performing logical operations. The LLM is trained in a few-shot learning setting, where it learns to generate these programs by observing a small set of examples. The program generator leverages the few-shot learning ability of Codex [4], a language model pre-trained on code.

For example, consider the claim: *"Both James Cameron and the director of the film Interstellar were born in Canada."* The reasoning program generated to verify this claim might look like the following:

```
def program():
    fact_1 = Verify("James Cameron was born in Canada.")
    answer_1 = Question("Who is the director of the film Interstellar?")
    fact_2 = Verify(f"{answer_1} was born in Canada.")
    label = Predict(fact_1 and fact_2)
```

In this program, the verification process is broken down into several steps:

1. **Verify James Cameron's birthplace**: The `Verify` function checks the truthfulness of the statement *"James Cameron was born in Canada."* and stores the result in `fact_1`.

2. **Identify the director of Interstellar**: The `Question` function answers *"Who is the director of the film Interstellar?"* and stores the response in `answer_1`.

3. **Verify the director's birthplace**: Using the answer from the previous step, the `Verify` function checks if that individual was born in Canada, storing the result in `fact_2`.

4. **Predict the overall claim's validity**: The `Predict` function combines the results of the previous verifications using a logical AND operation (`fact_1 and fact_2`) to determine the overall truthfulness of the original claim, storing the final verdict in `label`.

3.1.2 Program Execution

Once the reasoning program is generated, it is executed by an interpreter. To account for the fact that there may be multiple valid reasoning paths, ProgramFC generates a set of N candidate reasoning programs, where each program represents a unique chain of reasoning to verify the claim. After executing all N programs, the final verdict is determined by taking a majority vote over the predicted labels from each program. Each step in the program is delegated to a specialized function designed to handle that specific type of sub-task. For instance, a question-answering function might be used to answer questions posed in the program, while a fact verification function might be used to verify simpler claims. The results of each step are then used as input for subsequent steps, allowing for complex, multi-step reasoning. ProgramFC utilizes the FLAN-T5 pre-trained Transformer model [5] for these sub-tasks, fine-tuned on a small number of in-domain examples.

This program-guided approach offers several advantages. First, it enhances the explainability of the fact-checking process. The generated reasoning program serves as a transparent and human-readable explanation of how the model arrived at its verdict. Second, it improves data efficiency by leveraging the few-shot learning capabilities of LLMs. This means that ProgramFC can be trained on a relatively small amount of data, making it more practical for real-world applications where labeled data is scarce. Finally, ProgramFC is flexible, as the sub-task functions can be easily swapped or modified to adapt to different fact-checking scenarios or domains.

Analysis of ProgramFC

While ProgramFC presents a promising approach, we identify the following weaknesses it has:

The artificial nature of the reasoning program format may hinder its adoption by non-

technical users, who are typically unfamiliar with coding. This makes it difficult for general users to utilize this approach for verifying their desired claims. Furthermore, the need for a large number of examples to cover diverse reasoning types results in lengthy prompts, potentially affecting efficiency. The discrepancy between reasoning program grammar and standard programming languages can also pose challenges. Additionally, not all LLMs are specially trained on coding. Codex, for instance, is one of the few coding language models, a GPT language model fine-tuned on publicly available code from GitHub [4]. This limitation restricts users to running this approach on their desired LLM.

3.2 First-Order-Logic-Guided Knowledge-Grounded (FOLK) Reasoning

FOLK presents an alternative approach to automated fact-checking that also leverages the in-context learning power of LLMs. Unlike existing methodologies, which rely heavily on large-scale, human-annotated datasets for accurate verification, FOLK bypasses this dependency by retrieving relevant information from external sources like Google Search or Wikipedia. This allows FOLK to find evidence that can support or refute a claim, making it a more flexible solution in environments where human-annotated evidence may not be available. The key steps in FOLK are as follows:

3.2.1 FOL-Guided Claim Decomposition

FOLK translates the input claim into a First-Order Logic (FOL) clause, which is a formal representation of the claim's logical structure. This formulation consists of predicates, each representing a sub-claim that needs to be verified. This structured representation helps guide the LLM's reasoning process by breaking down complex

claims into more manageable components. For example, given the claim "Thomas Loren Friedman has won more Pulitzer Prizes than Colson Whitehead", FOLK would decompose this into predicates that correspond to verifiable sub-claims:

- Won(Thomas Loren Friedman, Pulitzer Prize) ::: Verify the number of Pulitzer Prizes Thomas Loren Friedman has won.
- Won(Colson Whitehead, Pulitzer Prize) ::: Verify the number of Pulitzer Prizes Colson Whitehead has won.

These predicates are then parsed into follow-up questions such as "How many Pulitzer Prizes did Thomas Loren Friedman win?" and "How many Pulitzer Prizes did Colson Whitehead win?" Each of these questions is subsequently answered using grounded knowledge from external sources. This decomposition into FOL predicates enables FOLK to systematically verify each sub-claim before arriving at a final conclusion about the main claim.

3.2.2 Knowledge Grounding

To ensure accurate claim verification, FOLK grounds its reasoning in reliable, external knowledge sources such as Google Search or Wikipedia. More specifically, FOLK applies a retriever mechanism for intermediate questions generated during the reasoning process, leveraging the top-1 result returned by Google Search through the Retrieval-Augmented Generation (RAG) technique. Answers obtained from Google Search are treated as knowledge-grounded, ensuring the LLM's verification process relies on concrete, up-to-date evidence. By using these sources, FOLK mitigates factual errors and prevents the LLM from generating hallucinations, all without requiring datasets annotated with evidence. This independence from annotated evidence distinguishes FOLK from traditional claim verification methods, enhancing its adaptability

in real-world scenarios.

3.2.3 Veracity Prediction and Explanation Generation

FOLK uses the FOL clauses and grounded knowledge to guide the LLM in determining whether the claim should be supported or refuted. It also produces a human-readable explanation to justify the decision. The quality of these explanations is manually evaluated based on three key properties: coverage, soundness, and readability. Additionally, the use of FOL predicates ensures that the LLM focuses on key aspects of the claim, which leads to more targeted and informative explanations.

FOLK's approach offers several advantages. First, it eliminates the need for annotated evidence, making it more scalable and applicable to a wider range of claims. Second, it provides transparent and human-readable explanations, which are crucial for building trust in automated fact-checking systems. Third, it leverages the reasoning capabilities of LLMs, allowing it to handle complex claims that require multi-hop reasoning. Additionally, FOLK has been shown to be effective even with smaller language models, demonstrating its potential for broader applicability.

3.3 First-Order-Logic-Decomposition and Cost-Effective Claim Verification (FOLDCoVe)

Building on the strengths of ProgramFC and FOLK, we propose a novel approach to automated fact-checking, called **F**irst-**O**rder-**L**ogic-**D**ecomposition and **Co**st-Effective Claim **Ve**rification (FOLDCoVe). FOLDCoVe aims to leverage the reasoning capabilities of LLMs while maintaining efficiency and explainability in the verification process.

3.3.1 Information Decomposition

Central to the FOLDCoVe framework is the concept of information decomposition. Given a complex claim, FOLDCoVe utilizes a powerful language model (PLM) to break it down into its constituent atomic propositions, or "atoms". These atoms represent the smallest self-contained units of meaning within the claim, each focusing on a single aspect. This decomposition is guided by FOL, a formal system for representing knowledge and reasoning.

The process of information decomposition involves three sequential steps:

Interpretation and Predicate Definition

The PLM analyzes the claim, identifying key entities, relationships, and concepts. It then defines predicates to represent these elements within the FOL framework.

FOL Translation

The claim is translated into an FOL formula using the defined predicates. This formal representation captures the logical structure of the claim, making it amenable to systematic reasoning.

Anglicization

The FOL formula is then "anglicized", or converted back into fluent English. This step ensures that the resulting atomic propositions are easily understandable by humans and maintain the semantic connections present in the original claim.

Prompt Engineering for FOL Decomposition

The prompt used for breaking down the claim into sub-claims is as follows:

Convert the input to first-order logic. Your output should consist of a set of predicates and a logical formula that combines them.

- Never miss a constant (entity) in the input that contributes to the formula.
- Replace variables with constants whenever possible.
- Avoid unary predicates solely for modeling constants.
- Avoid unused variables in the formula.

After producing the formula, anglicize it.

Example: {
input:
The logo of the American athletic shoe and clothing manufacturer who trademarked slogan coined by Dan Wieden has made $26 billion in profit.

predicates:
M(x): x is an American athletic shoe and clothing manufacturer.
S(x, y): x trademarked the slogan y.
C(y, z): Slogan y was coined by z.
P(x, w): The logo of the manufacturer x has made w profit.

formula:
Exists x, y (M(x) and S(x, y) and C(y, 'Dan Wieden') and P(x, '$26 billion'))

anglicized:
There exists an American athletic shoe and clothing manufacturer, let's call

it 'X', which has trademarked a slogan, let's call it 'Y'. This slogan 'Y' was coined by Dan Wieden. Additionally, the logo of this manufacturer 'X' has generated a profit of $26 billion.
}

input: <target_claim>

3.3.2 Verification

Once the claim is decomposed into its atomic propositions, or "sub-claims", FOLD-CoVe employs a weaker and more cost-effective language model (WLM) to verify the truth of each sub-claim. This verification process leverages external knowledge sources, such as Wikipedia or specialized knowledge bases, to assess the veracity of each sub-claim. The WLM analyzes the evidence from these sources and determines whether it supports or refutes the corresponding sub-claim. All these processes are done internally within the WLM in a single call per claim.

Contextual-Closed-Book setting

As language models continue to advance, stronger and larger models are being developed that often do not require explicit provision of evidence through external knowledge sources. In the Contextual-Closed-Book setting, these models rely on their internal parametric knowledge, pre-trained and fine-tuned on extensive textual corpora such as Wikipedia. Instead of retrieving evidence explicitly, the model is directed to focus on specific sections of its internal knowledge, like a designated source (e.g., Wikipedia), maximizing the use of pre-existing knowledge. This enhances the accuracy of the model's responses by narrowing its focus to a particular domain or source, reducing potential distractions from its broader training data. Our input

prompt for this setting is as follows:

Your task is to verify the following claim using your knowledge from Wikipedia:

<target_claim>

Instructions:

1. Research the different aspects of this claim on Wikipedia.
2. Write out all the relevant quotes from Wikipedia. (key: "evidence")
3. Do a step-by-step reasoning and analysis of the claim.
4. Write out your reasoning and explanation. (key: "reasoning")
5. Conclude if the claim is entirely true or false, with "True" or "False". If any part is unsupported, choose "False". Do not use "Mostly True" or "Partially True". The claim should only be marked "True" if every aspect is fully supported by the evidence.

Output format (in JSON):

```
{
  "evidence": "...",
  "reasoning": "...",
  "prediction": "True" or "False"
}
```

3.3.3 Comparison between FOLDCoVe and FOLK

While both FOLDCoVe and FOLK leverage the concept of first-order logic for language model prompt engineering in the context of fact-checking complex claims, they differ in several key aspects of their methodologies:

1. **Claim Decomposition:**

   - **FOLK** breaks down the claim into a set of FOL predicates, each representing a sub-claim to be verified. It then generates one question per predicate to assess its truth value. If all predicates evaluate to True, the claim is classified as SUPPORTED; otherwise, it's REFUTED. FOLK doesn't explicitly generate the full FOL formula, but implicitly assumes a conjunction of all predicates.
   - **FOLDCoVe** goes further by prompting the language model to provide both the FOL predicates and the complete FOL formula, allowing for more complex logical structures beyond simple conjunctions (e.g., using quantifiers and other logical connectives). Additionally, FOLDCoVe generates an "anglicized" (plain English) version of the FOL formula for better human readability, and it doesn't generate questions based on predicates.

2. **Language Model Usage:**

   - **FOLK** utilizes a single large language model for both claim decomposition and veracity prediction. This can lead to high computational costs, a limitation acknowledged in the FOLK paper itself.
   - **FOLDCoVe** adopts a more efficient approach inspired by RouteLLM [17], employing two LLMs: a powerful language model (PLM) for the initial claim decomposition and a weaker, more cost-effective language model

(WLM) for verifying the individual atomic propositions. This reduces the reliance on the computationally expensive PLM.

3. **Knowledge Grounding:**

   - **FOLK** explicitly grounds the LLM's answers in external knowledge sources, primarily using Google Search to retrieve relevant information for verification.
   - **FOLDCoVe** leverages the internal knowledge of the WLM, which has been pre-trained and fine-tuned on vast textual data (like Wikipedia). This allows FOLDCoVe to verify claims without explicitly querying external knowledge sources, further enhancing efficiency.

In summary, while both FOLDCoVe and FOLK utilize first-order logic to enhance LLM-based fact-checking, FOLDCoVe distinguishes itself through its more advanced claim decomposition, more efficient use of language models, and reliance on implicit knowledge grounding.

3.3.4 Advantages of FOLDCoVe

The FOLDCoVe framework offers several advantages over existing approaches:

Efficiency

By using a PLM only for the initial information decomposition step, FOLDCoVe minimizes the computational cost associated with powerful LLMs. Since claims are typically short, converting them into FOL is relatively inexpensive. However, verifying the FOL representation against large evidence texts can be resource-intensive with a PLM. By delegating this verification to the WLM, which is significantly more

cost-effective, FOLDCoVe efficiently handles the more extensive evidence required for the veracity check.

Additionally, the anglicization of the FOL formula not only enhances human readability but also allows for processing the entire claim with a single call to the WLM, further reducing computational costs. It is also worth mentioning that by binding predicates through shared variables, the model can efficiently verify interdependent sub-claims, ensuring a comprehensive and accurate assessment of the overall claim.

Explainability

The FOL-guided decomposition of the claim provides a clear and transparent explanation of the reasoning process. The anglicized FOL formula, along with the verification result, offers a human-readable justification for the final verdict.

Accuracy

By leveraging the strengths of both PLMs and WLMs, FOLDCoVe achieves a high level of accuracy in fact-checking. The PLM ensures accurate decomposition of complex claims, while the WLM focuses on verifying simpler propositions, leading to a more robust and reliable overall verification process.

Structure and Understanding

Rephrasing the claim in terms of FOL significantly enhances the model's comprehension. FOL's structure, akin to a database management system (DBMS), facilitates a clear mapping of claim components: FOL predicates resemble SQL tables, variables correspond to table columns, and constant values equate to column values. This transformation effectively converts unstructured textual data into a structured format, enabling the model to grasp the claim's intricate details and interrelationships

more effectively.

Chapter 4

Experiments

4.1 Overview and Statistics of the HOVER Dataset

4.1.1 Introduction to the HOVER Dataset

The HOVER (HOppy VERification) dataset is a large-scale fact extraction and verification dataset specifically designed to challenge models on multi-hop reasoning tasks [11]. It contains claims that require integrating evidence from multiple Wikipedia articles. The complexity of claims varies from two to four hops, where each "hop" represents the need for evidence from an additional Wikipedia article. The dataset aims to simulate real-world fact verification, where information must be aggregated from multiple sources, and it is used widely for testing models that perform multi-hop reasoning and fact-checking.

4.1.2 Definition of Hop

In the context of the HOVER dataset, a **hop** refers to a step in reasoning where the claim verification process requires extracting evidence from a different Wikipedia article. For example, in a two-hop claim, the model needs to retrieve information

from two different articles to substantiate the claim. As the number of hops increases, the reasoning complexity grows, requiring models to process longer chains of interconnected facts.

4.1.3 Dataset Collection Process

The HOVER dataset was created in three stages, leveraging the structure of the HOTPOTQA dataset [22]. Initially, 2-hop claims were generated by crowd-workers from HOTPOTQA question-answer pairs. These claims were validated by a separate group of annotators. To create 3-hop and 4-hop claims, entities in 2-hop claims were replaced with new information sourced from additional Wikipedia articles, effectively increasing the number of hops required to verify the claims. This process aimed to ensure that models would need to retrieve evidence from multiple documents, thereby simulating more complex reasoning scenarios.

The evidence retrieval process requires models to find supporting facts from various sections of Wikipedia articles, often utilizing hyperlinks between articles. This structure mimics how human fact-checkers would search for and connect information from multiple sources.

4.1.4 Dataset Statistics

The HOVER dataset consists of over 26,000 claims, which are split into training, validation, and test sets. These claims are further divided based on the number of hops required for verification:

- Two-hop claims: 1,126 claims in the validation set
- Three-hop claims: 1,835 claims in the validation set
- Four-hop claims: 1,039 claims in the validation set

The average length of the claims increases with the number of hops, making multi-hop claims inherently more complex. For instance, two-hop claims average around 19 tokens in length, while four-hop claims extend to 31.6 tokens on average.

4.2 Evaluation Metrics

In classification tasks, various metrics are used to evaluate the model's performance. These metrics include **Precision**, **Recall**, **Accuracy**, **F1 score**, and **F1 Macro Average**. Each metric is designed to measure different aspects of the model's effectiveness. Below are the definitions of these metrics:

Precision: Precision measures the proportion of correctly predicted positive instances among all instances predicted as positive. The formula is:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

Recall: Recall (also known as Sensitivity or True Positive Rate) quantifies the proportion of actual positive instances that were correctly identified by the model. The formula is:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

Accuracy: Accuracy measures the overall proportion of correctly predicted instances out of all instances. The formula is:

$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Instances (TP + TN + FP + FN)}}$$

F1 Score: F1 score is the harmonic mean of precision and recall, and it provides a single metric that balances both. The formula is:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1 Macro Average: F1 Macro Average calculates the F1 score by first computing the F1 score for each class individually and then averaging these scores across all classes. It gives equal weight to each class, making it especially useful when dealing with imbalanced datasets where the performance across all classes needs to be assessed fairly.

Furthermore, in binary classification — such as fact-checking, where the goal is to predict whether claims are true or false — the performance metrics **precision-0**, **precision-1**, **recall-0**, **recall-1**, **F1-0**, and **F1-1** are used to evaluate the model's effectiveness for each class individually. Typically, class 0 represents the negative class ("False"), and class 1 represents the positive class ("True"). The specific definitions for class 0 and class 1 are as follows:

Precision-0: Precision-0 (Precision for Class 0) measures the proportion of correctly predicted "False" claims among all claims predicted as "False". In other words, out of all claims the model predicts to be "False", how many are actually "False"?

Precision-1: Precision-1 (Precision for Class 1) measures the proportion of correctly predicted "True" claims among all claims predicted as "True". That is, out of all claims the model predicts to be "True", how many are actually "True"?

Recall-0: Recall-0 (Recall for Class 0) quantifies the proportion of actual "False" claims that were correctly identified by the model. In other words, out of all claims that are actually "False", how many did the model predict as "False"?

Recall-1: Recall-1 (Recall for Class 1) quantifies the proportion of actual "True" claims that were correctly identified by the model. That is, out of all claims that are actually "True", how many did the model predict as "True"?

F1-0: F1-0 (F1 Score for Class 0) represents the harmonic mean of precision and recall for "False" claims, balancing precision and recall.

F1-1: F1-1 (F1 Score for Class 1) represents the harmonic mean of precision and recall for "True" claims, providing a balance between these two metrics.

4.3 Performance in Closed-Book and Contextual-Closed-Book Settings

Table 4.1 presents a comparative analysis of FOLDCoVe and the state-of-the-art (SOTA) methods, ProgramFC and FOLK. For ProgramFC, we set N, the number of candidate reasoning programs, to 1 in order to maintain consistency with our single reasoning chain approach. Notably, ProgramFC, like FOLDCoVe, utilizes two models: one for program generation and another for program execution. In the table, the highest values across methods for the 2-hop, 3-hop, and 4-hop reasoning tasks are underlined.

Approach	**Language Model**	**HOVER (2-Hop)**	**HOVER (3-Hop)**	**HOVER (4-Hop)**
ProgramFC	GPT-4-Turbo + Mixtral-8x7B-Instruct	0.58	0.50	0.49
FOLK		0.57	0.50	0.48
FOLDCoVe		0.63	0.58	0.55
ProgramFC	GPT-4-Turbo + Llama-3-70B-Chat	0.60	0.57	0.52
FOLK		0.46	0.45	0.47
FOLDCoVe		<u>0.70</u>	<u>0.61</u>	<u>0.58</u>

Table 4.1: F1-MacroAvg scores for FOLDCoVe and the SOTA methods (Closed-Book setting)

The performance comparison of different language models on the Hover 2-hops, 3-hops, and 4-hops datasets in both the Closed-Book and Contextual-Closed-Book settings is presented in Tables 4.2, 4.3, 4.4 and Tables 4.5, 4.6, 4.7, respectively. In these experiments, the models incorporate GPT-4-Turbo as part of the FOLDCoVe approach to break down the claim into sentences.

In the context of our experiments, **FOL** refers to the proposed method's approach of decomposing and reformulating claims into logical forms, while **RAW** refers to the scenario where the original claims are directly utilized without additional processing. Additionally, in this work, we expand upon previous studies, which primarily reported F1-MacroAvg scores. We emphasize the importance of evaluating and reporting other metrics – including precision, recall, and accuracy – to provide a more comprehensive understanding of model performance. To highlight the best-performing models, we

have underlined the highest value for each metric across the respective datasets in the tables.

Language Model	**Mixtral-8x7B-Instruct**		**Llama-3-70B-Chat**	
Method	RAW	FOL	RAW	FOL
Precision-0	0.65	0.67	0.70	0.72
Recall-0	0.69	0.63	0.77	0.71
F1-0	0.67	0.65	0.74	0.72
Precision-1	0.61	0.60	0.70	0.67
Recall-1	0.56	0.64	0.62	0.69
F1-1	0.59	0.62	0.66	0.68
Accuracy	0.63	0.63	0.70	0.70
F1-MacroAvg	0.63	0.63	0.70	0.70

Table 4.2: FOLDCoVe results on Hover-2hops dataset (Closed-Book setting)

Language Model	**Mixtral-8x7B-Instruct**		**Llama-3-70B-Chat**	
Method	RAW	FOL	RAW	FOL
Precision-0	0.54	0.55	0.57	0.58
Recall-0	0.71	0.63	0.73	0.68
F1-0	0.61	0.59	0.64	0.62
Precision-1	0.64	0.62	0.67	0.66
Recall-1	0.46	0.53	0.49	0.55
F1-1	0.53	0.57	0.57	0.60
Accuracy	0.58	0.58	0.61	0.61
F1-MacroAvg	0.57	0.58	0.60	0.61

Table 4.3: FOLDCoVe results on Hover-3hops dataset (Closed-Book setting)

Language Model	**Mixtral-8x7B-Instruct**		**Llama-3-70B-Chat**	
Method	RAW	FOL	RAW	FOL
Precision-0	0.53	0.55	<u>0.59</u>	0.58
Recall-0	0.66	0.57	<u>0.70</u>	0.63
F1-0	0.59	0.56	<u>0.64</u>	0.61
Precision-1	0.54	0.55	<u>0.61</u>	0.58
Recall-1	0.41	0.53	0.49	<u>0.54</u>
F1-1	0.47	0.54	0.54	<u>0.56</u>
Accuracy	0.54	0.55	<u>0.60</u>	0.58
F1-MacroAvg	0.53	0.55	<u>0.59</u>	0.58

Table 4.4: FOLDCoVe results on Hover-4hops dataset (Closed-Book setting)

Language Model	**Mixtral-8x7B-Instruct**		**Llama-3-70B-Chat**	
Method	RAW	FOL	RAW	FOL
Precision-0	0.66	<u>0.71</u>	0.66	0.69
Recall-0	0.65	0.62	<u>0.84</u>	0.78
F1-0	0.65	0.66	<u>0.74</u>	<u>0.74</u>
Precision-1	0.60	0.61	<u>0.74</u>	0.70
Recall-1	0.61	<u>0.70</u>	0.51	0.59
F1-1	0.60	<u>0.65</u>	0.60	0.64
Accuracy	0.63	0.66	0.69	<u>0.70</u>
F1-MacroAvg	0.63	0.66	0.67	<u>0.69</u>

Table 4.5: FOLDCoVe results on Hover-2hops dataset (Contextual-Closed-Book setting)

Language Model	**Mixtral-8x7B-Instruct**		**Llama-3-70B-Chat**	
Method	RAW	FOL	RAW	FOL
Precision-0	0.54	0.54	0.55	0.57
Recall-0	0.67	0.58	0.82	0.74
F1-0	0.60	0.56	0.66	0.65
Precision-1	0.62	0.60	0.70	0.68
Recall-1	0.49	0.57	0.39	0.49
F1-1	0.55	0.58	0.50	0.57
Accuracy	0.57	0.57	0.59	0.61
F1-MacroAvg	0.57	0.57	0.58	0.61

Table 4.6: FOLDCoVe results on Hover-3hops dataset (Contextual-Closed-Book setting)

Language Model	**Mixtral-8x7B-Instruct**		**Llama-3-70B-Chat**	
Method	RAW	FOL	RAW	FOL
Precision-0	0.56	0.57	0.57	0.59
Recall-0	0.66	0.53	0.81	0.70
F1-0	0.61	0.55	0.67	0.64
Precision-1	0.57	0.55	0.65	0.62
Recall-1	0.46	0.59	0.37	0.50
F1-1	0.51	0.57	0.47	0.55
Accuracy	0.56	0.56	0.59	0.60
F1-MacroAvg	0.56	0.56	0.57	0.60

Table 4.7: FOLDCoVe results on Hover-4hops dataset (Contextual-Closed-Book setting)

From analyzing the tables 4.8, 4.9, and 4.10, we can derive several insights based on the ablation study comparing the performance of different models and settings (Closed-Book (CB) vs Contextual-Closed-Book (CCB)) for the HOVER dataset across different hop levels (2-hop, 3-hop, and 4-hop).

For the Llama-3-70B-Chat model, the CCB setting outperforms the CB setting in two evaluation metrics across all tables: precision-1 and recall-0. This indicates that, when using the CCB setting, the model is more accurate in identifying "True" claims (precision-1) and better at recognizing actual "False" claims (recall-0). Meanwhile, for the Mixtral-8x7B-Instruct model, CCB outperforms CB specifically in recall-1, meaning it shows a higher ability to correctly identify "True" claims out of all actual "True" claims. This suggests that the CCB setup can provide a more effective retrieval process, by restricting the model's internal knowledge to Wikipedia as the sole information source.

Language Model	**Mixtral-8x7B-Instruct**		**Llama-3-70B-Chat**	
Method	CB	CCB	CB	CCB
Precision-0	0.67	0.71	0.72	0.69
Recall-0	0.63	0.62	0.71	0.78
F1-0	0.65	0.66	0.72	0.74
Precision-1	0.60	0.61	0.67	0.70
Recall-1	0.64	0.70	0.69	0.59
F1-1	0.62	0.65	0.68	0.64
Accuracy	0.63	0.66	0.70	0.70
F1-MacroAvg	0.63	0.66	0.70	0.69

Table 4.8: FOLDCoVe ablation study on HOVER 2-hops dataset (Closed-Book vs. Contextual Closed-Book)

Language Model	**Mixtral-8x7B-Instruct**		**Llama-3-70B-Chat**	
Method	CB	CCB	CB	CCB
Precision-0	0.55	0.54	0.58	0.57
Recall-0	0.63	0.58	0.68	0.74
F1-0	0.59	0.56	0.62	0.65
Precision-1	0.62	0.60	0.66	0.68
Recall-1	0.53	0.57	0.55	0.49
F1-1	0.57	0.58	0.60	0.57
Accuracy	0.58	0.57	0.61	0.61
F1-MacroAvg	0.58	0.57	0.61	0.61

Table 4.9: FOLDCoVe ablation study on HOVER 3-hops dataset (Closed-Book vs. Contextual Closed-Book)

Language Model	**Mixtral-8x7B-Instruct**		**Llama-3-70B-Chat**	
Method	CB	CCB	CB	CCB
Precision-0	0.55	0.57	0.58	0.59
Recall-0	0.57	0.53	0.63	0.70
F1-0	0.56	0.55	0.61	0.64
Precision-1	0.55	0.55	0.58	0.62
Recall-1	0.53	0.59	0.54	0.50
F1-1	0.54	0.57	0.56	0.55
Accuracy	0.55	0.56	0.58	0.60
F1-MacroAvg	0.55	0.56	0.58	0.60

Table 4.10: FOLDCoVe ablation study on HOVER 4-hops dataset (Closed-Book vs. Contextual Closed-Book)

4.4 Performance in Gold-Evidence Setting

As shown in Table 4.11, FOLDCoVe consistently surpasses the SOTA methods in the gold-evidence setting across all instances. This highlights the effectiveness of our approach in leveraging the power of LLMs for reasoning over provided evidence. Furthermore, it's worth noting that the choice of language model significantly impacts performance. For example, in the gold-evidence setting, FOLDCoVe with Llama-3-70B-Chat and GPT-4-Turbo achieves the highest F1 scores, demonstrating the benefits of using a powerful, state-of-the-art language model.

Approach	**Language Model**	**HOVER (2-Hop)**	**HOVER (3-Hop)**	**HOVER (4-Hop)**
ProgramFC	GPT-4-Turbo + Mixtral-8x7B-Instruct	0.65	0.53	0.61
FOLK		0.64	0.54	0.51
FOLDCoVe		0.67	0.61	0.64
ProgramFC	GPT-4-Turbo + Llama-3-70B-Chat	0.66	0.61	0.62
FOLK		0.42	0.43	0.46
FOLDCoVe		0.74	<u>0.73</u>	<u>0.71</u>
ProgramFC	GPT-4-Turbo + GPT-3.5-Turbo	0.69	0.64	0.64
FOLK		0.59	0.52	0.52
FOLDCoVe		0.72	0.67	0.65
ProgramFC	GPT-4-Turbo + GPT-4-Turbo	0.71	0.68	0.68
FOLK		0.70	0.55	0.50
FOLDCoVe		<u>0.78</u>	0.72	0.70

Table 4.11: F1-MacroAvg scores for FOLDCoVe and the SOTA methods (Gold-Evidence setting)

All results in Tables 4.12, 4.13, and 4.14 are reported under the gold-evidence setting, utilizing the Hover 2hops, 3hops, and 4hops datasets respectively.

Language Model	Mixtral-8x7B-Instruct		Llama-3-70B-Chat		GPT-3.5-Turbo		GPT-4-Turbo	
Method	RAW	FOL	RAW	FOL	RAW	FOL	RAW	FOL
Precision-0	0.68	0.75	0.81	0.81	0.78	0.79	0.87	0.87
Recall-0	0.62	0.58	0.79	0.67	0.67	0.65	0.72	0.70
F1-0	0.65	0.65	0.80	0.74	0.72	0.71	0.79	0.77
Precision-1	0.60	0.61	0.76	0.68	0.67	0.66	0.72	0.70
Recall-1	0.66	0.77	0.79	0.82	0.79	0.80	0.86	0.87
F1-1	0.63	0.68	0.78	0.74	0.72	0.73	0.79	0.78
Accuracy	0.64	0.67	0.79	0.74	0.72	0.72	0.79	0.78
F1-MacroAvg	0.64	0.67	0.79	0.74	0.72	0.72	0.79	0.78

Table 4.12: FOLDCoVe results on Hover-2hops dataset (Gold-Evidence setting)

Language Model	Mixtral-8x7B-Instruct		Llama-3-70B-Chat		GPT-3.5-Turbo		GPT-4-Turbo	
Method	RAW	FOL	RAW	FOL	RAW	FOL	RAW	FOL
Precision-0	0.55	0.57	0.73	0.75	0.63	0.63	0.69	0.66
Recall-0	0.79	0.69	0.73	0.66	0.73	0.74	0.84	0.82
F1-0	0.65	0.63	0.73	0.70	0.68	0.68	0.75	0.73
Precision-1	0.69	0.66	0.76	0.72	0.72	0.72	0.82	0.80
Recall-1	0.43	0.54	0.76	0.80	0.62	0.61	0.66	0.63
F1-1	0.53	0.60	0.76	0.76	0.67	0.67	0.73	0.70
Accuracy	0.60	0.61	0.74	0.73	0.67	0.67	0.74	0.72
F1-MacroAvg	0.61	0.61	0.74	0.73	0.68	0.67	0.75	0.72

Table 4.13: FOLDCoVe results on Hover-3hops dataset (Gold-Evidence setting)

Language Model	Mixtral-8x7B-Instruct		Llama-3-70B-Chat		GPT-3.5-Turbo		GPT-4-Turbo	
Method	RAW	FOL	RAW	FOL	RAW	FOL	RAW	FOL
Precision-0	0.58	0.64	0.75	0.77	0.62	0.63	0.67	0.66
Recall-0	0.71	0.64	0.64	0.62	0.80	0.79	0.83	0.85
F1-0	0.64	0.64	0.69	0.68	0.70	0.70	0.74	0.75
Precision-1	0.61	0.63	0.68	0.67	0.70	0.70	0.77	0.78
Recall-1	0.47	0.63	0.78	0.80	0.49	0.52	0.59	0.55
F1-1	0.53	0.63	0.73	0.73	0.58	0.60	0.67	0.65
Accuracy	0.59	0.64	0.71	0.71	0.65	0.66	0.71	0.70
F1-MacroAvg	0.59	0.64	0.71	0.71	0.64	0.65	0.70	0.70

Table 4.14: FOLDCoVe results on Hover-4hops dataset (Gold-Evidence setting)

4.5 Analysis and Interpretation of Results

In our evaluation of various LLMs for fact-checking tasks, several key findings emerged regarding the sensitivity to prompt engineering and the effectiveness of different models.

4.5.1 Sensitivity to Prompt Engineering

In this study, by rephrasing the claim, we aimed to mathematize and anglicize it, making it well-structured and crystal clear to the language models, removing any ambiguity that they might have encountered in the original claim. Therefore, we expected any language model to perform better, or at least the same, when dealing with the new version of the claim compared to the original.

The Mixtral-8x7B-Instruct model showed significant sensitivity to prompt engineering. Its performance varied considerably based on how the prompts were structured. When the claim was rephrased to be clearer and more precise, Mixtral-8x7B-Instruct demonstrated noticeable improvement in its responses. In contrast, GPT models (GPT-3.5-Turbo and GPT-4-Turbo) demonstrated robust performance out-of-the-box, with minimal impact from changes in prompt structure. They performed exceptionally well without the need for extensive prompt optimization, maintaining their high level of understanding regardless of how the claim was presented.

We can explain our interpretation of these results using an analogy. Let us assume that all language models are students sitting in a classroom, and we, as human beings, are the teachers who try to explain concepts to them. Mixtral-8x7B-Instruct is like an average student, not having a very high IQ, whose progress is visible when the teacher expresses a concept (claim) in a better way. Whenever the teacher clarifies and structures the concept more effectively, this student seems to understand it bet-

ter, showing considerable improvement in performance; we might call this student an *A student* due to their significant progress.

On the other hand, there are highly smart and talented students in the class, such as GPT-3.5-Turbo and GPT-4-Turbo, who grasp the concepts on the first attempt. Their level of understanding and reasoning is already so high that they do not need extra clarification. They do not make any impressive progress because they are already performing at an excellent level, consistently maintaining their performance; we might call them *A+ students*.

However, there are some students — such as Llama-3-70B-Chat — who have a huge knowledge base but do not seem to improve or function any better when the teacher elaborates on the concept. Sometimes, additional explanations might even cause them to perform worse. These students do not exhibit strong reasoning and learning abilities; we might refer to them as *B students*.

This analogy highlights the varying degrees of sensitivity to prompt engineering among different language models. While models like Mixtral-8x7B-Instruct can significantly benefit from well-crafted prompts, others like GPT-3.5-Turbo and GPT-4-Turbo perform exceptionally well without such optimization. This underscores their utility in practical applications where prompt engineering might not be feasible.

4.5.2 Impact of FOL

The application of FOL to reformulate complex claims into simpler, more natural statements provided notable improvements in classification accuracy. For Mixtral-8x7B-Instruct, FOL-enhanced claims resulted in up to an 8.5% increase in F1-MacroAvg score, with average increases of 1.8% in the Closed-Book setting, 1.6% in Contextual Closed-Book, and 4.4% in Gold-Evidence. For Llama-3-70B-Chat, our approach boosted the F1-MacroAvg score by an average of 4.5% in the Contextual Closed-

Book setting but did not lead to noticeable improvements in the Closed-Book and Gold-Evidence settings.

4.5.3 Improving Precision and Recall

FOL techniques were particularly beneficial for optimizing the precision of class 0 (False) and the recall of class 1 (True). By rephrasing claims, FOL helped the models to more accurately identify false claims and better recall true claims, thus enhancing overall performance.

4.5.4 Contexual-Closed-Book Process

Given that Wikipedia content is part of the training data for all the LLMs used in our study, it was observed that providing pieces of evidence together with the claim was quite unnecessary. The LLMs inherently performed an internal mini RAG process, utilizing their training data to generate responses. This capability significantly simplifies the input requirements and leverages the models' pre-existing knowledge effectively.

These results underscore the varying needs for prompt engineering across different models and highlight the substantial benefits of using FOL for claim verification tasks. Additionally, the inherent RAG abilities of LLMs reduce the need for external evidence input, streamlining the fact-checking process.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we have explored the development and evaluation of fact-checking systems powered by LLMs, focusing on improving accuracy, efficiency, and explainability in the verification process. Our research, grounded in recent advancements in natural language processing, introduced novel methods that address the complexity of verifying claims that require multi-step reasoning and external evidence.

The key contributions of this work are as follows:

1. We proposed FOLDCoVe, a novel fact-checking framework that combines FOL with LLMs to decompose complex claims into manageable sub-claims, enhancing both explainability and verification accuracy. This decomposition allows for more efficient fact-checking by delegating simpler verification tasks to a weaker, cost-effective language model (WLM), thereby minimizing computational costs.

2. FOLDCoVe demonstrated that the integration of FOL into fact-checking tasks enables improved comprehension of claim structures. By rephrasing claims in logical form, FOLDCoVe helps LLMs understand intricate relationships within

claims, which resulted in higher accuracy, particularly when working with multi-hop reasoning tasks.

3. Our framework introduced the concept of anglicizing FOL formulas, making the logical representation of claims more human-readable. This not only enhances transparency and explainability but also facilitates easier debugging and interpretation of the model's decision-making process.

4. The comparison between FOLDCoVe and other state-of-the-art methods highlighted significant improvements in both efficiency and accuracy. FOLDCoVe 's innovative approach of using two different LLMs for claim decomposition and verification proved to be more computationally efficient than methods relying solely on a powerful language model for all steps of the fact-checking process.

The results from our experiments underscored the potential of LLMs, particularly when combined with structured methodologies like FOL, to handle complex fact-checking tasks. FOLDCoVe consistently outperformed existing methods in terms of both efficiency and accuracy across a variety of fact-checking scenarios, including multi-hop reasoning tasks, demonstrating its versatility and robustness.

In conclusion, this thesis offers substantial advancements in the design of automated fact-checking systems, contributing to both the theoretical and practical understanding of how LLMs can be effectively employed to ensure the veracity of complex claims. Through the development of FOLDCoVe, we provide a scalable, explainable, and efficient solution to the growing demand for automated fact-checking in today's information-driven world.

5.2 Future Work

The current implementation of FOLDCoVe, while promising, offers several avenues for further exploration and improvement. One potential direction is to refine the verification process. Instead of directly anglicizing the FOL formula, each predicate could be treated as a "sub-query" to be verified independently by the WLLM. This would involve instantiating the variables within each predicate and querying the knowledge base for relevant information. The verification results for each predicate could then be aggregated, potentially using a many-valued logic system that considers the probabilities or confidence scores associated with each sub-query. This approach could lead to a more nuanced and accurate assessment of the overall claim's veracity.

Additionally, this approach allows for the parallelization of the variable instantiation and knowledge grounding process. Since the verification of each sub-query is independent of the others, these steps could be performed in parallel, potentially leading to significant speedups in the overall fact-checking process. This would be particularly beneficial when dealing with large volumes of claims or claims with a large number of atomic propositions.

Bibliography

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] AI@Meta. Llama 3 model card. 2024.

[3] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[5] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[6] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*, 2023.

[7] Herbert B Enderton. *A mathematical introduction to logic*. Elsevier, 2001.

[8] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[9] Ivan Ilin. Advanced rag techniques: an illustrated overview, 2023.

[10] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[11] Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. Hover: A dataset for many-hop fact extraction and claim verification. *arXiv preprint arXiv:2011.03088*, 2020.

[12] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[13] A Koubaa. Gpt-4 vs. gpt-3.5: A concise showdown. 2023.

[14] Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. Ra-dit: Retrieval-augmented dual instruction tuning. *arXiv preprint arXiv:2310.01352*, 2023.

[15] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*, 2023.

[16] Ggaliwango Marvin, Nakayiza Hellen, Daudi Jjingo, and Joyce Nakatumba-Nabende. Prompt engineering in large language models. In *International conference on data intelligence and cognitive informatics*, pages 387–402. Springer, 2023.

[17] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

[18] Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. Fact-checking complex claims with program-guided reasoning. *arXiv preprint arXiv:2305.12744*, 2023.

[19] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

[20] Haoran Wang and Kai Shu. Explainable claim verification via knowledge-grounded reasoning with large language models. *arXiv preprint arXiv:2310.05253*, 2023.

[21] Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. Search-in-the-chain: Interactively enhancing large language models with search for knowledge-intensive tasks. In *Proceedings of the ACM on Web Conference 2024*, pages 1362–1373, 2024.

[22] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.