Towards Reliable Fact-Verification Systems

Bachelor Thesis
Faculty of Science, University of Bern

submitted by

Deniz Demir

from Murten, Switzerland

Supervision:

PD Dr. Kaspar Riesen
Corina Masanti
Institute of Computer Science (INF)
University of Bern, Switzerland

Abstract

In an era defined by the rapid spread of digital misinformation, the ability to verify factual claims automatically has become increasingly important. This study presents the development and evaluation of a fast, modular fact-checking pipeline that combines real-time evidence retrieval from Google Search with a fine-tuned transformer model, specifically Mistral-7B, for claim verification. The system takes claims as input, converts them into questions, retrieves relevant information from Google, and classifies the evidence as either supporting, refuting, or insufficient using the fine-tuned model. To improve the classification accuracy, especially for the task of detecting refuted claims, the base model is fine-tuned on a self-created synthetic, task-specific dataset. The pipeline is evaluated using another self-created synthetic dataset, GPT2500, as well as two known fact-checking benchmarks, FEVER and SciFact. Results show that fine-tuning significantly improves recall and F1-scores for the `REFUTES` class, addressing a key weakness of the instruction-tuned base model. Furthermore, this study presents a breakdown of the performance by snippet source, revealing a critical mismatch between retrieved evidence and evidence quality, especially in organic search results. Despite these advances, several limitations remain, ranging from incorrect retrieval outputs to the challenges of fact-checking subtle misinformation. Nevertheless, the results highlight the potential of combining web search with LLM adaptation to enable scalable fact-checking across a wide range of topics. This work contributes both a functional prototype and an empirical foundation for future research in automated claim verification.

Acknowledgements

I am grateful to be able to write my thesis in the pattern recognition group of the University of Bern, as the diverse work they do really interests me. Furthermore, I would like to thank PD Dr. Kaspar Riesen and Corina Masanti for their help in guiding and structuring my approach, which has helped me a lot. Finally, I would like to thank my partner and my family for always supporting me.

Contents

Chapter 1

Introduction

Artificial Intelligence (AI) has become a central technology of our time, improving systems that influence how we communicate, make decisions, and interact with the world. As AI continues to advance, the capacity to process and generate human language improves with it [1]. Natural Language Processing (NLP) enables machines to understand, interpret, and produce text and speech in a way that was previously unimaginable. This advancement has opened new paths for different industries, from virtual assistants and chatbots to machine translation, summarization, and automated fact-checking [2]. Within the larger context of AI, this study is situated at the intersection of NLP, machine learning, and information retrieval. Figure 1.1 shows how the topic of this study fits into the context of AI and the related subfields. It also shows how NLP is closely related to areas such as textual entailment, information retrieval, text generation, and model fine-tuning. These are the components that come together in this work, which focuses on building a fast and robust fact-checking pipeline that is capable of verifying claims using real-time web evidence.

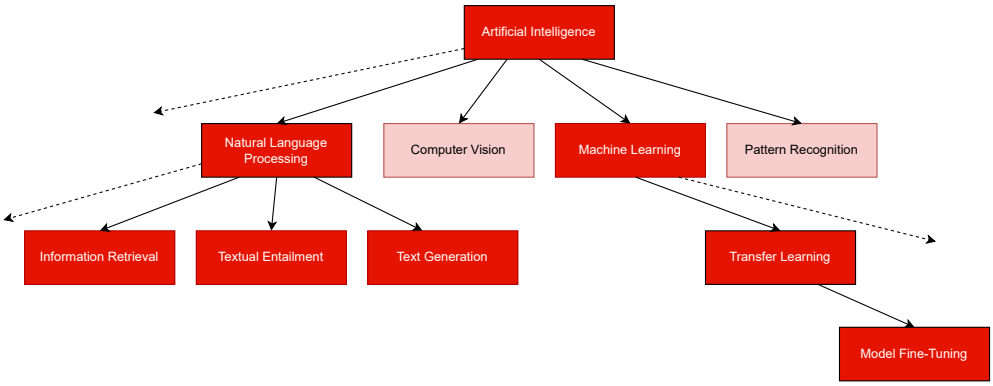

Figure 1.1: The topic of the present study in a broader context.

The need for automated fact-checking systems is more relevant than ever. In the digital space, which is flooded with information, consumers are often confronted with conflicting or false information, especially in areas such as news, politics, or social media discussions [3]. Manual fact-checking of these claims is often extremely time-consuming, and the resources of professional fact-checkers are often limited. Therefore, researchers are trying to automate these systems to make the fact-checking process more responsive to real-time information [4].

This study contributes to that movement by building a modular pipeline, which performs claim verification based on Google Search engine result page (SERP) evidence. The central idea is to take a claim as input, query the web for relevant evidence, and determine whether the evidence supports, refutes, or is insufficient to evaluate the claim. While this sounds conceptually straightforward, it raises several challenges, such as retrieving meaningful evidence automatically, accurately mapping it to the original claim, and reasoning over potentially incomplete information.

At the core of the system is a large language model (LLM), specifically Mistral-7B, which is further fine-tuned to the task of claim verification. Even though pre-trained models are also able to understand general-purpose language, their performance on specific tasks such as fact-checking can be limited without further adaptation [5]. Fine-tuning, which is the process of continuing model training on a task-specific dataset, has proven effective to address this limitation [6]. Therefore, one of the core objectives of this study is to investigate whether fine-tuning a pre-trained model can lead to meaningful performance gains in real-world fact-checking scenarios.

The research question that guides this work is the following:

How effective is a fast fact-checking pipeline combining Google Search-based evidence retrieval and a fine-tuned LLM (Mistral-7B) for claim verification?

This question is central to the technical and scientific parts of this project. The pipeline is designed to be fast, while using search engine results instead of static databases to verify claims. It retrieves snippets from Google's SERP, converts them into natural questions, and passes the retrieved evidence with the original claim to a fine-tuned LLM, which determines whether a statement is true, false, or lacks sufficient information. The goal is to build a system that works in an open-domain setting and can handle a wide range of topics without relying on static sources such as Wikipedia. The focus lies on detecting false information, as finding this is one

of the most important and challenging aspects of fact-checking.

To evaluate the pipeline, this study uses standard fact-checking benchmarks, specifically FEVER and SciFact, and a custom synthetic dataset (GPT2500), which is created for this project. This allows for a detailed analysis of the system's performance. Furthermore, this study analyzes how the source quality of the retrieved evidence affects the quality and the accuracy of the predictions. This provides valuable insights into the limitations of the system and where improvements are needed.

Furthermore, the ability to detect false information automatically has an impact on journalism, education, and daily life in general. A system that can reliably verify claims at scale could serve as a fact-checking assistant, which would help users to discern truth from misinformation. Even if this study does not solve the problem fully, it proposes a promising approach and presents empirical evidence.

In sum, this study explores how the recent advancements in LLMs and web-based information retrieval can be used to build an effective and efficient fact-checking pipeline. It offers both a practical implementation and a structured evaluation of the system's components, which contribute to the goal of developing an intelligent tool to combat misinformation in the digital age.

In terms of structure, the thesis is structured as follows. Chapter 2 lays the theoretical base by introducing the core concepts of false claim detection, Google Search results, transformer models, fine-tuning language models, and related work in the field of automated fact-checking. Chapter 3 presents the methodology, where the pipeline is explained in detail. This includes each step of the pipeline, synthetic data generation, as well as fine-tuning, which are key parts of this study. Chapter 4 presents and explains the empirical results of the performance analysis across datasets, labels, and evidence sources. To conclude, Chapter 5 provides a critical reflection of the empirical approach while discussing the strengths and weaknesses of the method. Additionally, possible future work is discussed.

Chapter 2

Theoretical Background and Foundations

This chapter elaborates on the theory behind the fact-checking pipeline and puts it into context. It also focuses on NLP and fine-tuning language models. NLP is the key component of the built pipeline, since the aim of this study is to develop a system that can automatically verify information in text by comparing it with evidence gathered from the web. NLP is a subfield of AI, focused on enabling computers to understand, interpret, and generate human language. One of the main parts of NLP is text classification, which involves categorizing text based on its content to ultimately label the claims as supported, refuted, or not providing sufficient information [7].

2.1 False Claim Detection

False claim detection is a central task in automated fact-checking, with the goal to determine the truthfulness of statements based on external information. It is typically regarded as a supervised classification problem, where the system assigns a label to a claim based on its truthfulness [8]. The most commonly used labels are `SUPPORTS`, `REFUTES`, and `NOT ENOUGH INFO` [9]. These labels indicate whether the evidence supports the claim, contradicts it, or does not provide enough information for a conclusion.

The theoretical basis for false claim detection comes from semantic textual entailment, a subfield of NLP, which determines whether the meaning of one text segment can be inferred from another. In the sense of false claim detection, the claim is treated as a hypothesis, and the evidence as the premise. The model then assesses whether the premise logically entails, contradicts, or provides insufficient information to evaluate the hypothesis. In the present study, false claim detec-

tion is the primary objective of the developed fact-checking pipeline. The system processes the input text, retrieves evidence from Google's SERP, and labels each claim-evidence pair using a fine-tuned transformer model. The model then outputs one of the three mentioned labels.

Benchmark datasets have driven research in claim verification by providing standardized evaluation frameworks. One of the most influential is the Fact Extraction and VERification (FEVER) dataset, which is a standard benchmark for this task [9]. The FEVER dataset is a large-scale benchmark where systems are required to retrieve the relevant information from Wikipedia and classify whether it supports the claim, refutes it, or if there is insufficient evidence to label the claim [9]. Another relevant dataset is SciFact, which adapts claim verification to biomedical claims [10]. Furthermore, the LIAR dataset offers claim-level annotations based on journalistic fact-checks [11]. However, many existing datasets for claim verification operate within constrained environments, meaning they rely on fixed evidence such as Wikipedia. While this controlled setup makes the evaluation easier, it does not reflect the messiness of real-world information retrieval. In contrast, the developed pipeline is designed to function in an open-domain setting, retrieving evidence dynamically from the web. For this reason, this study additionally uses a custom dataset, which better reflects real-world scenarios.

2.2 Google Search Results

Search engine results are the primary entry to web-based information retrieval and play a crucial role in the evidence gathering stage of automated fact-checking systems [12]. The outputs returned by search engines such as Google are structured into different result types, including featured snippets, AI-generated overviews, knowledge graphs, and the organic search results. Each of these results contains a different degree of structure, reliability, and completeness, and their differences have a direct impact on how they can be used for fact-checking. What follows is a description of the Google SERP components used to retrieve evidence in this study.

Organic Results

For Google, the organic search results usually consist of a URL, title, and a short snippet extracted from the page. These snippets are often not complete sentences, since they are displayed to give an overview of the page so the reader gets some information without the need to visit the page. The pages displayed after a request are ranked by algorithms designed to calculate the relevance and quality of a web

page with respect to the user's query. While the exact algorithm is not openly available, the general principle is to prioritize pages that are both authoritative and contextually relevant to the query. Search engine optimization is often used to improve these rankings naturally by structuring the content in a way that influences the ranking factors positively.

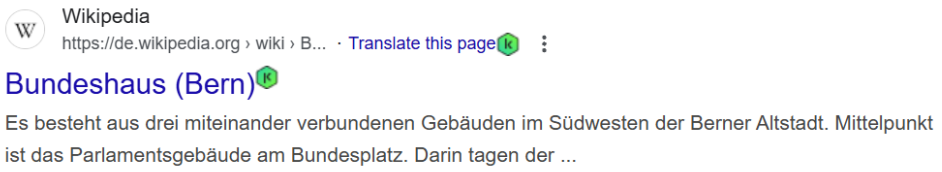

Figure 2.1: Example of an organic search result by Google.

Featured Snippets

Google's featured snippets are designed to answer the user's question directly. Using NLP, relevant information is taken from highly ranking webpages and directly displayed for the user. This structure makes them particularly valuable in this pipeline, since the claim verification stage benefits from evidence that is both contextually rich and concise. [13].

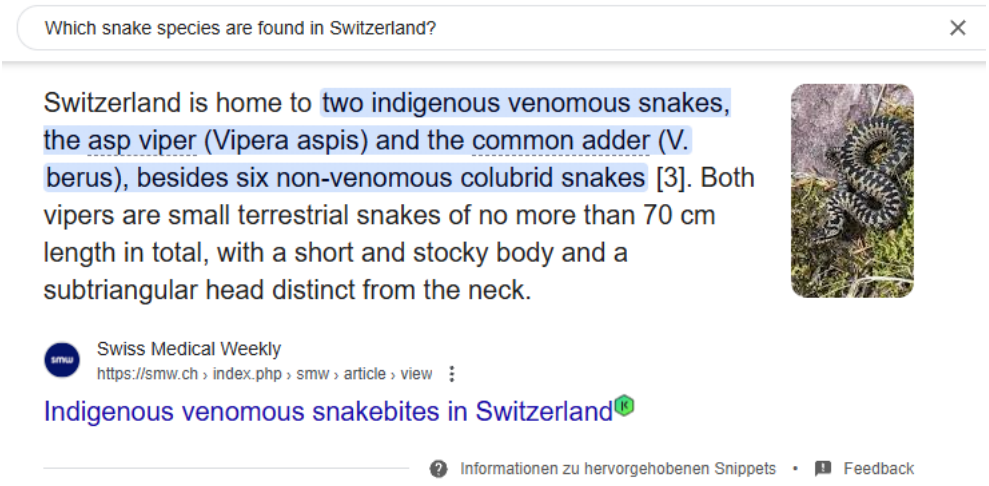

Figure 2.2: Example of a featured snippet by Google.

AI Overview

Another key part of Google's SERP is the AI Overview, the company's newest advancement in the search experience [14]. Google uses LLMs to summarize the content of high-ranking webpages regarding a query. These summaries consist of multiple sources. The goal of AI Overviews is to display a compact and accurate response without the need to click a page [15]. AI Overviews introduce both opportunities and complications for automated fact-checking. On one hand, they display relevant information that is compatible with claim-evidence comparison. On the other hand, they bring known limitations of generative models, such as hallucination and lack of attribution [16].

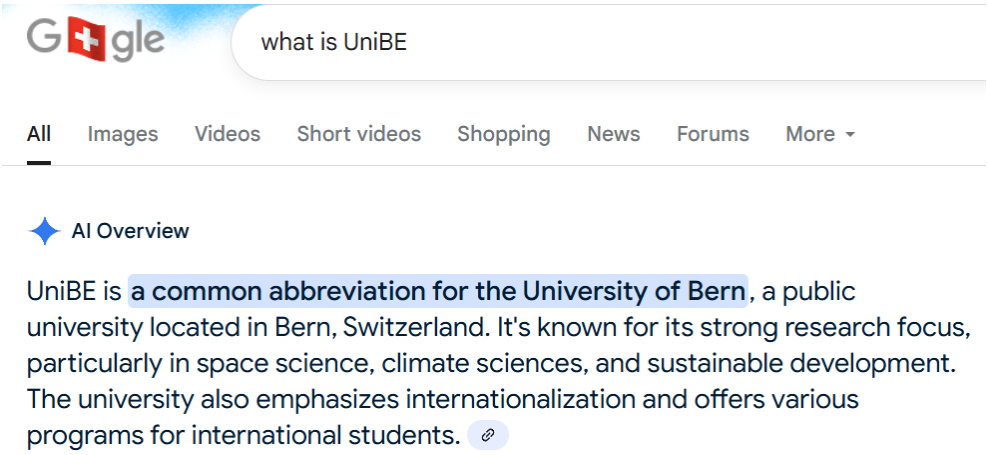

Figure 2.3: Example of an AI overview by Google.

Knowledge Graph

Another important part of Google's search experience is the knowledge graph. Knowledge graphs appear for well-known entities, for example, public figures, locations, and historical events. They provide highly structured content, including numbers, facts, and links, in addition to traditional search results. Google takes the relevant information from structured databases, including proprietary knowledge bases and public datasets. The information is presented in bullet points or tabular formats. Knowledge graphs are reliable sources to verify entity-level claims [17].

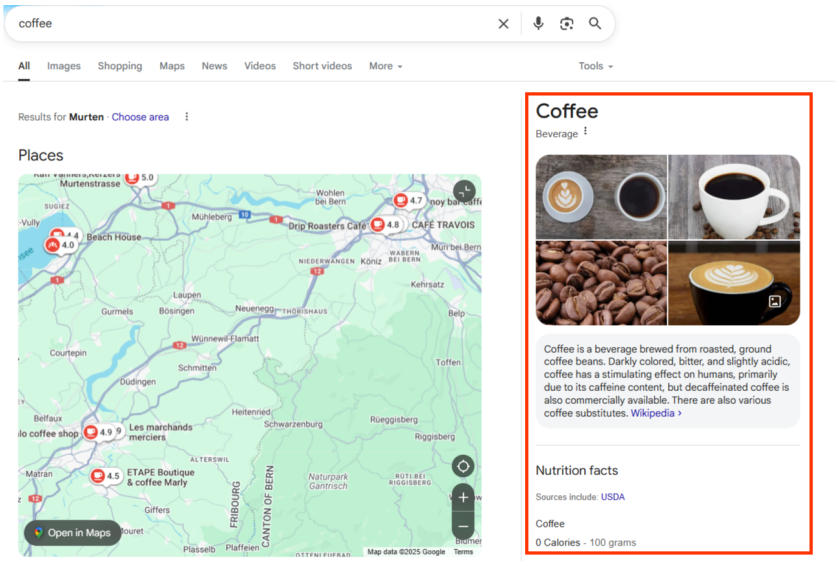

Figure 2.4: Example of a Knowledge Graph by Google.

2.3 Transformer Models

Transformer models are a foundational technology for modern LLMs, including those used in this study. Their architecture has significantly changed and continues to shape NLP. They were first introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017 [18]. Transformer models replace recurrence with attention mechanisms, which enable parallel computation and greatly improve scalability compared to earlier sequence models such as recurrent neural networks (RNNs) or long short-term memory networks (LSTMs).

Figure 2.5 shows the transformer architecture, with the encoder on the left and the decoder on the right. While attention mechanisms were used before in RNNs, the transformer architecture introduced self-attention as a core component, which allows processing entire sequences at once.

The most important part of the transformer architecture is the attention mechanism, which enables the model to compute the relationships between all tokens in a sequence at the same time. The key part is the scaled dot-product attention, which is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Here, Q, K, and V are the query, key, and value matrices, which are used to measure

how much attention each word should pay to the others. The term d_k refers to the size of the key vectors and is used to scale the attention scores.

This mechanism allows the model to look at how each word is connected to others in a sentence, which improves its ability to capture context. To better recognize different types of word relationships, the transformer applies multi-head attention, where multiple attention patterns are learned in parallel. The results are then combined to give a better understanding of the sentence. Each head focuses on a different part of the sequence. Furthermore, the original transformer model has two key components:

- An encoder, which processes the input and produces contextual embeddings.
- A decoder, which generates the output sequence one token at a time using masked attention.

Each block consists of attention layers, feed-forward layers, residual connections, and layer normalization.

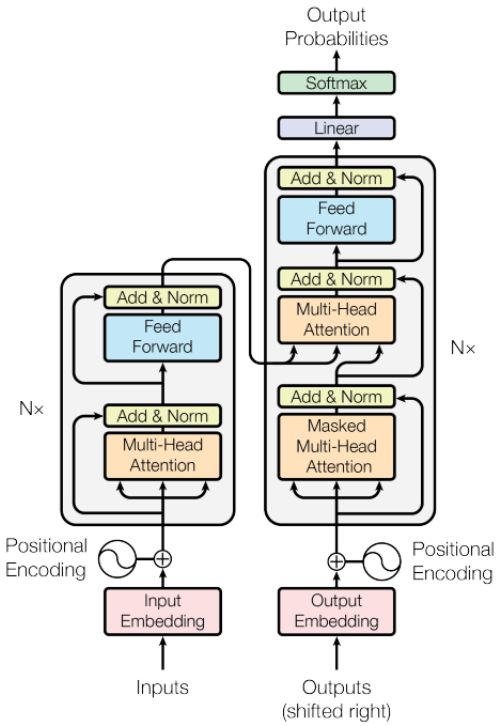

Figure 2.5: Overview of the Transformer architecture from Vaswani et al. [18], showing the encoder and decoder layers with multi-head attention and feed-forward networks.

To allow the model to learn sequence order, positional encodings are added to the input embeddings, either through sine functions or through learned vectors. This allows the attention mechanism to process text in a meaningful sequence. Recent

developments adapted the transformer into different variants tailored to specific tasks. Primarily, these include:

- BERT introduced masked language modeling with encoder-only architecture for understanding tasks [19].
- GPT introduced decoder-only autoregressive modeling for generation tasks [20].

This study uses decoder-only transformers, specifically Mistral-7B and GPT-4o, which are designed for causal language modeling and autoregressive generation. Autoregressive generation is a decoding approach in which the model predicts the next token in a sequence using only the previously generated tokens. This step-by-step process continues until the desired text is produced.

2.3.1 Mistral-7B

Mistral-7B is a decoder-only model launched by Mistral AI in 2023. With 7 billion parameters, it achieves comparable performance to much larger transformer models such as LLaMA-2 13B while keeping the model fast and efficient during training and use [21]. The Mistral-7B architecture contains a stack of decoder blocks, but introduces two key changes that enhance its efficiency:

- Sliding Window Attention (SWA) restricts the model's attention to a fixed window of nearby tokens in each layer. This helps to reduce memory usage while still allowing the model to build an understanding of longer contexts across layers [22].
- Grouped Query Attention (GQA) is a method where multiple attention heads use the same keys and values but have separate queries. This helps the model to run more efficiently without losing its ability to recognize different types of relationships in the text [22].

Mistral uses causal masking in the attention layers so that each token can only "see" previous tokens, which is important for generating text in order. Each layer of the model includes multi-head attention, feed-forward networks, residual connections, and normalization. It uses a byte-pair encoding (BPE) tokenizer to split the text into smaller pieces and can process up to 8,192 tokens at the same time [23].

In this study, Mistral-7B is used in two parts:

1. The instruction-tuned model (Mistral-7B Instruct) is used to convert claims into questions.

2. A fine-tuned version is used as a three-way classification model that compares the claim with the given evidence to label each one as `SUPPORTS`, `REFUTES`, or `NOT ENOUGH INFO`.

2.3.2 GPT-4o

While Mistral-7B is the central model used in the pipeline, a second model, GPT-4o, is used to generate synthetic data. GPT-4o, which stands for generative pretrained transformer 4 omni, was introduced by OpenAI in May 2024. It is also a decoder-only transformer model, similar to Mistral-7B, and is trained using causal language modeling, where the next token is predicted based on the preceding tokens using causal masking. GPT-4o is multimodal, meaning it can not only process text but also images and audio [24]. Despite this, in the context of this study, GPT-4o is only used in text mode to generate labeled training and validation data in JSON format.

Compared to earlier models such as GPT-4 or GPT-4 Turbo, GPT-4o offers significantly improved latency, efficiency, and response quality. It also uses Reinforcement Learning from Human Feedback (RLHF), a training method that uses ranked human preferences to improve the model [25]. Using GPT-4o to create synthetic data reflects a growing trend in NLP research: the use of LLMs as generative labeling engines to create supervised datasets at scale, especially in domains where data is rare or expensive to obtain [26]. Thanks to its strong generative capabilities and efficiency, GPT-4o is particularly well-suited for creating large-scale synthetic labeled datasets.

2.4 Fine-Tuning Language Models

LLMs such as BERT, GPT, or Mistral achieve strong performance due to pretraining on vast amounts of text using self-supervised learning [27]. LLMs use transfer learning to improve accuracy and performance, typically through a pretraining phase followed by fine-tuning. To improve their performance in specific downstream tasks, such as classification, summarization, or question answering, they are typically fine-tuned on task-specific data.

Fine-tuning is the process of adapting a pretrained model to a task-specific dataset. The model's parameters are updated so it can perform better on the new task, while keeping the language understanding obtained during pretraining [27]. Fine-tuning uses loss functions, such as cross-entropy for classification. This means that the model learns by comparing its predictions to the correct answers. This

process is often significantly faster and more data-efficient than training a model from scratch [28].

There are several standard paradigms for fine-tuning:

- Full fine-tuning, where all the model's parameters are updated. While this leads to strong performance, it uses a lot of resources and is usually not done for large models [6].
- Feature-based fine-tuning, where the pretrained model is frozen and only a task-specific classifier head is trained on top of the output representations [29].
- Parameter-Efficient Fine-Tuning (PEFT), a family of methods that updates only a small subset of the model's parameters while keeping the rest frozen. This is especially important for modern LLMs with billions of parameters [28].

Numerous studies show that fine-tuned models consistently outperform prompt-based zero-shot or few-shot approaches on well-defined tasks. For example, in fact verification tasks such as FEVER [10] and SciFact [30], models that are fine-tuned on claim-evidence pairs show improved precision and consistency in classification, compared to models used via prompting.

Overall, fine-tuning remains an important technique in modern NLP, combining general-purpose capabilities with focused performance, and enabling pretrained transformer models to adapt efficiently to new tasks with high accuracy and efficiency. In this work, fine-tuning is used to improve the model's ability to detect refuted claims.

2.5 Related Work

Automated fact-checking is an important application of NLP, especially with the rise of online misinformation, conspiracy theories, and politically motivated disinformation. Numerous systems have been proposed and discussed to automate this process, typically by breaking the task down into a pipeline of sub-tasks.

A survey by Zeng (2021) formulates automated fact-checking as a pipeline task, which consists of three main stages: claim detection, evidence retrieval, and claim validation. The survey states that an effective fact-checking system usually has all three stages. Zeng explains that the effectiveness of automated fact-checking systems depends heavily on two factors: the precision of the retrieved evidence and the robustness of the claim validation [3]. The second key point of reference is the FEVER benchmark by Thorne et al. (2018), which is one of the most

influential contributions in automated fact-checking research [9]. The pipeline developed by Thorne et al. describes fact-checking as a task of labeling claims as `SUPPORTED`, `REFUTED`, or `NOT ENOUGH INFO` based on the evidence extracted from Wikipedia. Their pipeline architecture has three steps: document retrieval, sentence selection, and natural language inference (NLI). The dataset itself contains over 185,000 claims, making it one of the largest resources for supervised learning in fact-checking. The FEVER framework has inspired follow-up studies, which focus on improving evidence retrieval. Furthermore, the structured nature of FEVER provides a valuable controlled setting for testing verification models, even if the reliance on Wikipedia limits its applicability to open-domain and real-time fact-checking scenarios.

Together, these two works represent important directions in automated fact-checking research: structured benchmark-driven model evaluation, and the broader integration of language models with search and reasoning components. They underline the importance of high-quality datasets, efficient retrieval methods, and robust classification systems when developing pipelines that aim to assess the truthfulness of claims at scale. However, many existing approaches remain limited by their reliance on static sources like Wikipedia, and do not fully address the challenges of open-domain, real-time fact verification. The proposed approach addresses this gap by combining dynamic web-based evidence retrieval with fine-tuned language models.

Chapter 3

Methodology

This chapter outlines the design of the fact-checking pipeline. It begins with an overview of the developed system, followed by an explanation of each processing step. The chapter then describes the generation of synthetic data, which serves two purposes in this work: fine-tuning the model to improve the detection of false claims, and providing a controlled dataset for evaluating the performance of the pipeline.

3.1 Fact-Checking Pipeline

The fact-checking pipeline processes text through a series of steps, from sentence splitting to claim verification. The final version of the pipeline is structured as follows:

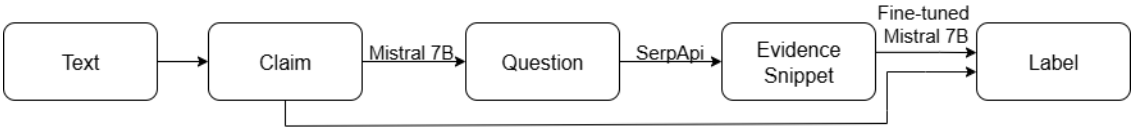

Figure 3.1: Structure of the final pipeline.

3.1.1 Sentence Splitting

The first step in the fact-checking pipeline is the splitting of text into sentences. To achieve this, the pipeline uses the free, open-source Python library SpaCy with the English model "en_core_web_sm". SpaCy uses the dependency parse to determine sentence boundaries. A dependency parse is a syntactic analysis of a sentence that represents the grammatical relationships between words as a set of directed links, showing which word depends on others [31]. This step is necessary since the pipeline

needs single sentences or claims to process in the next step. Once the text is split into single sentences, they are saved to a JSONL file for further processing.

In the following code fragment, the SpaCy library and its small English model en_core_web_sm is loaded and the function `process_text` is defined in Python.

```
import spacy
nlp = spacy.load("en_core_web_sm")
# Function to process text and split it into sentences
def process_text(input_text):
    doc = nlp(input_text)
    return [sent.text.strip() for sent in doc.sents]
```

The function takes the string input_text as input and splits it into sentences. The function `process_text` returns a list of sentences, removing any leading or trailing spaces. In the pipeline, the sentences are saved in JSONL format and are each labeled as a claim.

3.1.2 Question Generation

In this step, the claims are converted into questions. This is highly impactful, since Google Search responds better to questions, with regard to fast evidence finding, than simple statements. The claims are converted into questions using Together AI's Mistral Instruct LLM, which is an instruct fine-tuned version of Mistral-7B. Instruction fine-tuning is a process in which a language model is trained on datasets consisting of instructions and corresponding responses, enabling it to better follow user prompts and perform tasks in a conversational or task-oriented manner. [32] The following function `generate_question` takes each claim as input, and converts it into a question.

```
def generate_question(claim):
    prompt = f"""
    Change the following factual statement into a natural-sounding,
    ↪ grammatically correct question.
    Statement:
    "{claim}"
    Question:
    """
    try:
        response = client.chat.completions.create(
            model="mistralai/Mistral-7B-Instruct-v0.1",
            messages=[{"role": "user", "content": prompt}],
            max_tokens=100,
            temperature=0.7,
            top_p=0.9,
        )
        ...
```

The function sends the prompt to the Together AI API, requesting a completion from the LLM Mistral-7B Instruct. The parameters used in the API call are the following: Max_tokens=100 limits the length of the generated response to 100 tokens. This ensures that the generated question does not become excessively long or complex. Temperature=0.7 controls the creativity of the model's response. Lower values (close to 0) lead to more deterministic and focused outputs, and higher values (up to 2) produce more diverse and creative responses. The chosen value of 0.7 provides a balance between coherence and diversity, which proves effective for generating meaningful questions. The final parameter top_p=0.9, also known as nucleus sampling, limits the selection of tokens to those that make up the top 90% of the probability distribution. This helps to restrict the choices to the most likely, but still allows for diversity.

3.1.3 Evidence Retrieval

In this phase of the pipeline, the JSON file containing questions is processed with the Google Search API SerpApi to find evidence. SerpApi is a real-time API service that allows developers to access and extract structured data from search engine result pages (SERPs), such as Google, Bing, and others [33]. The goal is to extract the best possible answer, so in this pipeline, the API is used to extract one of four possible Google Search results with descending priority. The possible results are as follows: 1. AI Overview, 2. Featured Snippet, 3. Knowledge Graph, 4. Organic Result. By ranking the possible evidence sources, SerpApi can be used to extract the best available evidence for a given question. The evidence is categorized by source and saved as "snippet" with the original question in JSON format for further use.

3.1.4 Validation

To validate the claim with the found snippet, the pipeline uses two versions of Mistral-7B. One is the instruction-tuned base version, and the second is a fine-tuned version. First, the instruction-tuned base model (Mistral-7B Instruct) is used to establish a performance baseline. Then, the same model is further fine-tuned on task-specific data to evaluate whether the additional training improves the performance on claim verification. Each pair of claim and snippet is tokenized and loaded into the model. The model then predicts whether the claim is supported (`SUPPORTS`), refuted (`REFUTES`), or if there is insufficient information (`NOT ENOUGH INFO`). After every claim-snippet pair is processed, the predictions are saved as "predicted_label" in the JSON file for further analysis. Further information on the fine-tuned Mistral-7B can be found in Section 3.3 Mistral Fine-Tuning.

3.2 Dataset Creation - GPT2500

To evaluate the performance of the pipeline, a synthetic dataset is created. The goal is to generate short, natural-sounding statements across ten different topics - Health & Medicine, Climate & Environment, History, Politics & Policy, Economics, Technology & AI, Nutrition & Food Science, Psychology, Space & Astronomy, and Law & Rights. These topics are deliberately chosen for their complexity and diversity, making the dataset particularly challenging and suitable to test the robustness of the fact-checking pipeline. The dataset contains four primary classes: *True* statements, *Obvious* misinformation, *Subtle* misinformation, and *Very Subtle* misinformation. The dataset is generated using the OpenAI GPT-4 model (version: gpt-4o-2024-08-06) via the OpenAI API [34]. For the generation, different prompts and parameters are used for each misinformation category. To balance the diversity, temperature settings are changed for each class. For the *True* statements, which consist of verifiable, widely accepted, and stylistically varied sentences, a temperature between 0.4 and 0.6 is used to balance factual consistency. The *Obvious* misinformation category is designed to consist of clearly false but grammatically correct statements. For this category, a temperature of 0.8 is used to encourage the model to take more creative or varied approaches when generating text. The *Subtle* misinformation class consists of claims that appear true but contain nuanced errors such as imprecise terminology or subtly incorrect implications. For this category, a temperature of 0.9 is used to allow for richer stylistic diversity. For the *Very Subtle* misinformation category, a temperature of 0.9 is used as well. Additionally, five different stylistic patterns are chosen: implicit logical leaps, context collapse, misleading clarifications, swapped causality and statistical sleight. Implicit logical leaps are statements that lead the reader to draw an incorrect conclusion based on true or partially true premises. Context collapse means that the statements are technically true but misleading due to neglect of the context. Misleading clarification means that the sentences appear accurate but include a subtle contradiction or factual twist. Swapped causality sentences contain reversals of cause and effect that preserve grammatical plausibility, and statistical sleight means that factually correct numerical data is used in misleading or manipulative framing. The goal of this misinformation category is to push the pipeline to its limit and show the limitations of this approach. Each entry in the dataset contains the following data fields:

Field	Description
`claim`	The generated statement (either factual or containing misinformation).
`label`	Binary classification label: either `"True"` or `"False"`.
`misinformation_type`	One of `"Obvious"`, `"Subtle"`, `"Very Subtle"`, or `null` (for true statements).
`misinformation_style`	Applicable only to Very Subtle Misinformation; one of five predefined styles (e.g., `context_collapse`, `swapped_causality`).
`topic`	One of the ten predefined domains (e.g., `Health & Medicine`, `History`).
`id`	Unique identifier for the data entry.

Table 3.1: Metadata fields for each dataset entry.

The dataset is stored in JSON format to allow efficient parsing and batch processing. An entry is structured as follows:

```
{
  "id": 2326,
  "claim": "The Industrial Revolution caused the Agricultural Revolution as the
  ↪  need for more efficient farming techniques arose to support the growing
  ↪  urban population.",
  "label": "False",
  "misinformation_type": "Very Subtle",
  "misinformation_style": "swapped_causality",
  "topic": "History"
}
```

The GPT2500 dataset enables the evaluation of the fact-checking pipeline across different misinformation categories and topics.

3.3 Mistral Fine-Tuning

To improve the performance of the claim verification, the instruction-tuned Mistral-7B model is further adapted for a classification task by fine-tuning it on labeled claim-evidence pairs. A classification head is added, and the model is trained to predict one of three labels: `SUPPORTS`, `REFUTES`, or `NOT ENOUGH INFO`. Fine-tuning is done using a parameter-efficient approach, combined with model quantization to reduce computational cost.

3.3.1 Synthetic Data Generation

To generate training data for fine-tuning the model on the classification task in a controlled setting, OpenAI's GPT API is used. Data samples are generated

from the same ten topics as the GPT2500 dataset to keep it diverse. Each entry consists of a claim, a snippet of supporting or refuting evidence, a label that is either `SUPPORTS`, `REFUTES`, or `NOT ENOUGH INFO`, and a source type that is either answer_box_snippet or organic_result to mimic the evidence gathered by the SerpApi. The answer_box_snippet mimics the AI Overview and the Answer Box result by Google, while the organic_result mimics the default snippet retrieved by the SerpApi, which is the description of the first page regarding the search. This snippet is usually cut off in the middle of the sentence. The following prompt is used to generate the data in the required format:

```
def build_prompt(topic, label):
    return f"""
    Generate 5 realistic fact-checking training samples for the topic: {topic}.
    Each sample must include:
    - A claim
    - A short snippet that either supports or refutes it
    - A label: {label}
    - A source: 'answer_box_snippet' or 'organic_result'
    ...
    Respond with ONLY a valid JSON array of 5 objects. No explanations, no
    ↪ markdown.
    """
```

The training dataset consists of 3,800 `SUPPORTS`, 6,000 `REFUTES` and 3,600 `NOT ENOUGH INFO` samples. The class distribution is intentionally imbalanced to underscore the importance of detecting refuting evidence. This is a key function in fact-checking systems. As a result, the `REFUTES` class is oversampled to ensure the model is strongly exposed to a diverse range of negative examples. To prepare the data as input for the model, a preprocessing function merged the claim and snippet fields into a single string and applied truncation and padding:

```
def preprocess(example):
    text = f"Claim: {example['claim']} Evidence: {example['snippet']}"
    inputs = tokenizer(text, truncation=True, padding="max_length",
    ↪ max_length=256)
    label_map = {"SUPPORTS": 0, "REFUTES": 1, "NOT ENOUGH INFO": 2}
    inputs["label"] = label_map.get(example["label"])
    return inputs
```

The final dataset is split up into a training and a validation dataset following a 90% / 10% split.

3.3.2 LoRA-Based Adaptation

Due to the size of the Mistral-7B model, fine-tuning is done using Low-Rank Adaptation (LoRA) combined with 4-bit quantization. This approach ensures that only

a small subset of parameters needs to be updated [35]. With this setup, less GPU memory is required. The LoRA settings are as follows:

```
lora_config = LoraConfig(
    r=32, # The rank of the low-rank matrices inserted into the linear layers
    lora_alpha=32, # A scaling factor applied to the LoRA updates
    lora_dropout=0.22, # Dropout rate applied to the LoRA layers to improve
    ↪ generalization
    target_modules="all-linear", # Specifies that LoRA should be applied to all
    ↪ linear layers in the model
    use_rslora=True, # Enables rank-scaling
    bias="none", # Indicates that no biases should be adapted; only weights are
    ↪ modified
    task_type="SEQ_CLS" # Declares that the task is sequence classification,
    ↪ allowing the LoRA system to apply task-specific logic
)
```

The base model is loaded with the Hugging Face Transformers library [36]. Additionally, 4-bit quantization is used. This process approximates 16-bit floating-point numbers with a 4-bit representation, which allows the model to be loaded and fine-tuned on much weaker hardware.

```
quant_config = BitsAndBytesConfig(
    load_in_4bit=True, #Enables 4-bit loading
    bnb_4bit_use_double_quant=True, #Applies nested quantization for higher
    ↪ precision
    bnb_4bit_quant_type='nf4', #Uses "NormalFloat4", a robust quantization type
    bnb_4bit_compute_dtype=torch.float16 #Uses 16-bit floating point for
    ↪ intermediate computations
)
```

The base model is loaded with this quantization configuration and prepared for LoRA-based fine-tuning using the Hugging Face PEFT (Parameter-Efficient Fine-Tuning) library, which provides implementations of LoRA and other parameter-efficient techniques for LLMs. PEFT refers to a family of methods that fine-tune only a small subset of a model's parameters, significantly reducing computational cost while maintaining performance [37].

```
model = AutoModelForSequenceClassification.from_pretrained(
    model_ckpt,
    quantization_config=quant_config,
    num_labels=3,
    device_map="auto"
)
```

The PEFT library wraps the model and inserts LoRA layers while keeping most of the base model frozen in place. The training is done using the Hugging Face

Trainer API. After each epoch, checkpoints are saved and based on validation loss, the best-performing model is kept and exported. This training strategy combined efficient adaptation with memory optimization, making it well-suited for the Nvidia GeForce RTX 3070 Ti GPU used.

Chapter 4

Experimental Evaluation

This chapter evaluates the performance of the fact-checking pipeline introduced in Chapter 3. The goal is to evaluate its ability to correctly classify claims based on the given evidence. This is done with three different benchmark datasets, as well as two validation configurations: the base Mistral-7B Instruct with a prompt and the same model fine-tuned for the three-way claim verification presented in Section 3.3. This evaluation provides insights into the effect of fine-tuning and the correctness of the predictions of the fact-checking pipeline.

4.1 Evaluation Datasets

To ensure a complete evaluation, the pipeline is tested on three datasets with varying characteristics. In the following table, S, R, and NEI stand for `SUPPORTS`, `REFUTES`, and `NOT ENOUGH INFO` respectively.

Dataset	**Domain**	**Purpose / Focus**	**S**	**R**	**NEI**
FEVER	Wikipedia	Tests general classification	1,075	400	525
SciFact	Scientific literature	Specific scientific knowledge	331	173	–
GPT2500	Mixed (synthetic)	Varying difficulty levels	1,000	1,500	–

Table 4.1: Overview of evaluation datasets and label distribution.

The first dataset used, FEVER (Fact Extraction and VERification), is a well-known and widely used fact-checking benchmark, consisting of 185,000 claims that were manually written based on Wikipedia content. Each claim is given a label (`SUPPORTS`, `REFUTES`, or `NOT ENOUGH INFO`) and evidence sentences. From this dataset, the first 2,000 claims are used to evaluate the pipeline.

The second dataset used to evaluate the pipeline is SciFact. This is a domain-

specific dataset targeting scientific fact-checking. The claims were taken from scientific papers, primarily in the biomedical domain. In addition to the claim, supporting or refuting evidence was extracted from research abstracts and added to the dataset. This dataset introduces additional linguistic and logical complexity due to the technical nature of its content. The possible labels are `SUPPORTS` and `REFUTES`.

Lastly, the GPT2500 dataset is used to evaluate the pipeline on different levels of complexity, as explained in Section 3.2. This dataset also serves as the representative case for testing the pipeline.

These datasets include a wide range of language styles and claim types, which allows for a diverse evaluation of how well the system performs in realistic and synthetic scenarios. They are publicly available, including retrieved evidence and model outputs for both the base and the fine-tuned model (see Appendix B: Datasets).

4.2 Setup

The evaluation is done to compare two versions of the Mistral-7B model with the fact-checking pipeline:

Base Model

The instruction-tuned Mistral-7B Instruct, used in a zero-shot setup. For each claim-evidence pair, the following prompt is used:

```
def build_prompt(claim, snippet):
    return f"""
You are a fact-checking assistant. You will be given a claim and a piece of evidence (a snippet).
Determine whether the evidence SUPPORTS, REFUTES, or provides NOT ENOUGH INFO to verify the claim.

Respond with one of the following labels only: SUPPORTS, REFUTES, NOT ENOUGH INFO.

Claim: "{claim}"
Evidence: "{snippet}"

Label:
""".strip()
```

The model outputs are stored in JSON format for further evaluation.

Fine-Tuned Model

The same Mistral-7B model is used, which is further fine-tuned on a synthetic dataset of labeled claim-evidence pairs as described in Section 3.3. This version

uses a classification head and directly predicts one of the three labels without the need for a prompt.

For both models, the rest of the pipeline remains unchanged. The same set of input samples is used across the evaluations to ensure equal comparison. The output labels are saved to the input dataset and compared against the ground truth to compute the evaluation metrics.

All experiments are conducted on a machine with a Nvidia GeForce RTX 3070 Ti GPU (8GB VRAM), 32GB RAM, and an AMD Ryzen 7 7800X3D 8-Core CPU. Additionally, quantized model loading and parameter-efficient fine-tuning are enabled to make the experiment possible under these hardware constraints.

4.3 Evaluation Metrics

To evaluate the performance of the classification models, the standard multi-class evaluation metrics are used: precision, recall, F1-score, and macro-averaged F1-score. These metrics are chosen because it is a multi-class task and there is a class imbalance. While accuracy can be misleading in these scenarios, precision, recall, and the F1-score offer a more detailed view of the model's performance for each class individually.

A one-vs-all (OvA) approach is used. For each class (`SUPPORTS`, `REFUTES`, and `NOT ENOUGH INFO`), the model's predictions are evaluated as if that class is the positive label while the other two are considered negative. This allows for the computation of class-wise precision, recall, and F1-score. The formulas used are as follows:

Precision

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c}$$

Precision measures the proportion of correct positive predictions for a given class c. It answers the question: out of all the examples predicted as class c, how many predictions are actually correct? High precision indicates that the model is not over-predicting a class.

Recall

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c}$$

Recall measures how many of the actual examples of class c are correctly identified by the model. It answers the question: out of all the true instances of class c, how

many does the model capture? High recall means that the model is effective at capturing all relevant instances.

F1-Score

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

F1-score is the harmonic mean of precision and recall for class c. It provides a single score that balances both false positives and false negatives. This is especially useful when the class distribution is imbalanced, which is often the case in fact-checking.

To report a single, overall measure of model performance, the class-wise F1-scores are macro-averaged:

$$\text{Macro-averaged F1-score} = \frac{1}{C} \sum_{c=1}^{C} \text{F1}_c$$

Where $C = 3$ is the number of classes. This approach treats all classes the same, regardless of how often they appear in the data, ensuring that the performance of minority classes is properly reflected.

4.4 Results

The experimental evaluation focuses on two main objectives:

1. To assess the general effectiveness of the fact-checking pipeline across datasets from different domains.

2. To examine the performance improvement gained through fine-tuning the Mistral-7B model compared to its base instruction-tuned version.

To achieve this, the pipeline is evaluated on three datasets: FEVER, SciFact, and GPT2500. These datasets cover different domains and structures, allowing for a varied evaluation. However, to avoid redundancy and maintain clarity, only the GPT2500 dataset is presented in detail and serves as the representative case while the FEVER and SciFact datasets are shown and explained briefly. The complete evaluation results for the FEVER and SciFact datasets can be found in the Appendix.

4.4.1 GPT2500

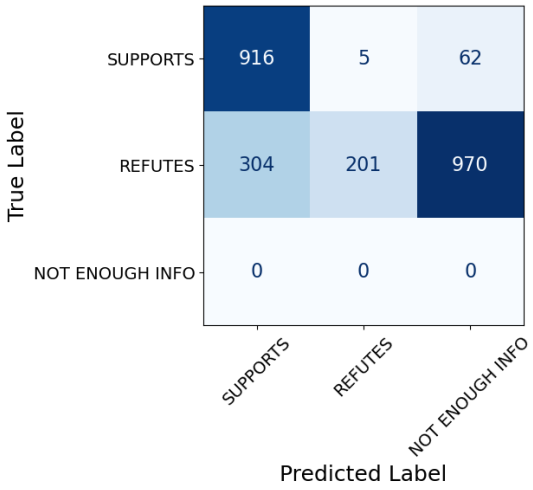

Figure 4.1: Confusion matrix for GPT2500 before fine-tuning.

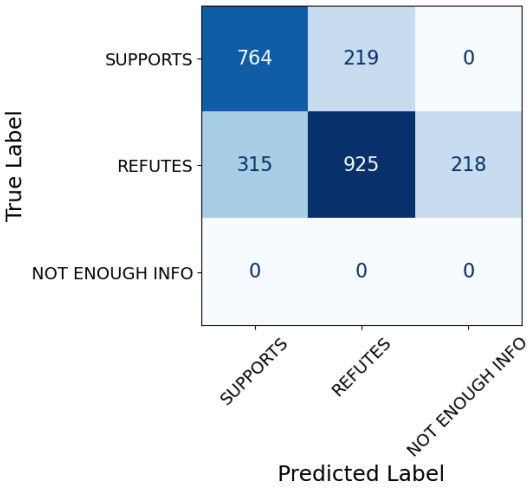

Figure 4.2: Confusion matrix for GPT2500 after fine-tuning.

Figures 4.1 and 4.2 show the confusion matrices for the GPT2500 dataset before and after fine-tuning. With the base model, which is seen in Figure 4.1, `SUPPORTS` samples are classified relatively well, with 916 correctly identified, but a large portion of the 1,475 `REFUTES` samples are misclassified. Specifically, 970 are labeled as `NOT ENOUGH INFO` and 304 as `SUPPORTS`, which indicates a strong bias of the base model towards predicting `NOT ENOUGH INFO`. After fine-tuning, Figure 4.2, the number of correctly identified `REFUTES` samples increases substantially to 925, demonstrating a much better ability to detect refuting evidence. However, this improvement came at a trade-off: the number of correctly classified `SUPPORTS` samples decreases from 916 to 764, with 219 now being misclassified as `REFUTES`. This trade-off suggests that while fine-tuning improves the detection of refuting evidence, it introduces new confusion between `SUPPORTS` and `REFUTES` classes.

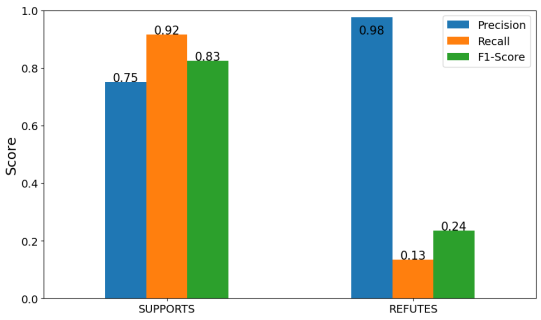

Figure 4.3: Evaluation metrics for GPT2500 before fine-tuning.

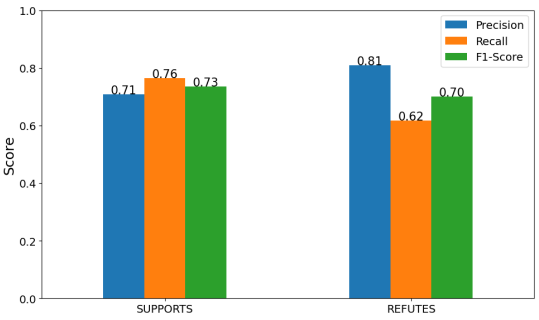

Figure 4.4: Evaluation metrics for GPT2500 after fine-tuning.

This trend is further visible in the class-wise precision, recall, and F1-score shown in Figures 4.3 and 4.4. Before fine-tuning, the model performs strongly on `SUPPORTS`, achieving a high recall of 0.92 and F1-score of 0.83. However, the performance of the `REFUTES` class is extremely poor, with a recall of 0.13 and an F1-score of just 0.24, despite a high precision of 0.98. This reflects a strong bias towards labeling claims as supportive, and an uncertainty in identifying misinformation.

After fine-tuning, the performance across both classes becomes more balanced. The F1-score of the `REFUTES` class increases substantially to 0.70, with recall improving to 0.62, indicating improved sensitivity to refuting evidence. The performance on `SUPPORTS` remains stable, with an F1-score of 0.73, which is a slight decrease that suggests a trade-off in favor of making more cautious predictions. This reduction is mainly due to a drop in recall from 0.92 to 0.76. After fine-tuning, the model is less willing to classify a claim as `SUPPORTS` without strong evidence, which increases `REFUTES` detection but slightly reduces correct `SUPPORTS` prediction.

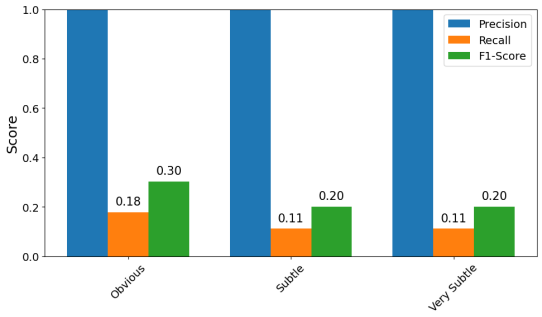

Figure 4.5: Detailed `REFUTES` performance for GPT2500 before fine-tuning.

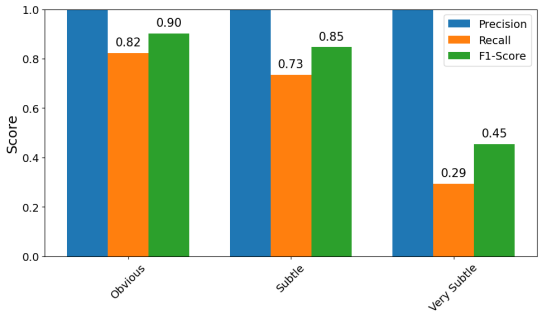

Figure 4.6: Detailed `REFUTES` performance for GPT2500 after fine-tuning.

Figures 4.5 and 4.6 provide a detailed look at the model's ability to detect refuted claims, broken down by misinformation type: *Obvious*, *Subtle*, and *Very Subtle*. These plots focus exclusively on claims whose ground truth label is `REFUTES`, and report the model's precision, recall, and F1-score for each subgroup.

Before fine-tuning, the model shows extremely low recall across all categories, particularly for *Subtle* and *Very Subtle* misinformation, indicating that it fails to identify the majority of refuted claims. It is important to note that the precision is misleading in these plots since only truly refuted claims are evaluated and therefore false positives are excluded. After fine-tuning, the performance improves a lot across all misinformation types. Recall for *Obvious* misinformation rises from 0.18 to 0.82, and the corresponding F1-score improves from 0.30 to 0.90. *Subtle* misinformation shows similar results with the F1-score going from 0.20 to 0.85. Even the *Very Subtle* misinformation class sees a significant increase in F1-score from 0.20 to 0.45. These plots show that with increasing subtleness, the difficulty for the model to

predict correctly increases as well.

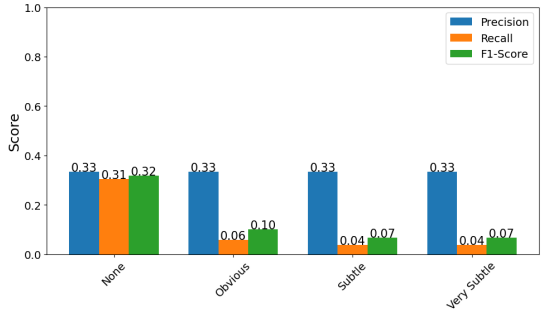

Figure 4.7: Macro-averaged F1-score for GPT2500 by misinformation type before fine-tuning.

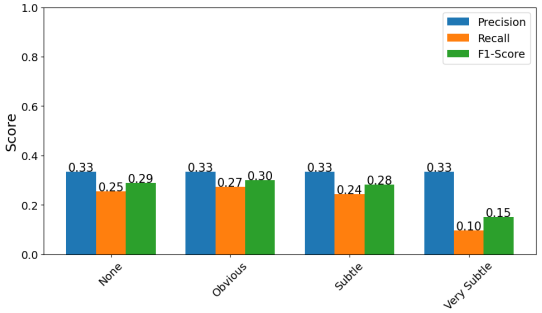

Figure 4.8: Macro-averaged F1-score for GPT2500 by misinformation type after fine-tuning.

In contrast to the previous analysis, which focuses exclusively on refuted claims, Figures 4.7 and 4.8 show the macro-averaged precision, recall, and F1-score for all three classes, `SUPPORTS`, `REFUTES`, and `NOT ENOUGH INFO`, grouped by misinformation type. Misinformation type *None* means the claim is true and there is no misinformation present. Macro-averaging calculates each score independently for the three classes and averages them, giving equal weight to each label regardless of whether there is a class imbalance. This provides a more balanced view of overall classification performance.

From the results indicate that the overall performance is relatively consistent across the *None*, *Obvious*, and *Subtle* classes, with macro F1-scores ranging from 0.28 to 0.30 after fine-tuning. However, the *Very Subtle* category, even after fine-tuning, shows only a moderate improvement in macro-averaged F1-score from 0.07 to 0.15. Similarly, the recall rises from 0.04 to 0.10. This confirms the earlier observation that with more subtle misinformation, the model's ability to predict correctly decreases. It is important to note that since this plot includes true claims as well, the performance differences reflect not only the model's ability to detect misinformation, but also its ability to support accurate claims.

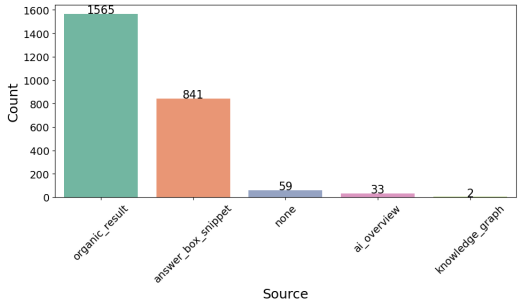

Figure 4.9: GPT2500 snippet count by source.

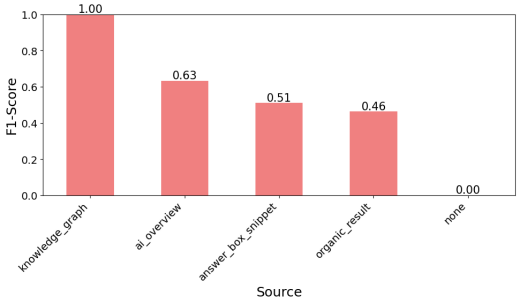

Figure 4.10: F1-score by source after fine-tuning.

Another important analysis is the distribution of snippet sources, which is shown in Figure 4.9. Figure 4.10 shows the corresponding F1-score for each source after fine-tuning. When compared directly, a clear mismatch comes to light. The most frequently retrieved source, which is the organic result with 1,565 samples, delivers the weakest results, reaching an F1-score of only 0.46 after fine-tuning. In contrast, sources such as *ai_overview* and *knowledge_graph*, which are rarely retrieved, perform the best. This shows that retrieval quantity does not equal retrieval quality. This is a clear bottleneck in the overall effectiveness of the pipeline and will be further discussed in Chapter 5.

4.4.2 FEVER

To complement the main evaluation, the pipeline is tested on the FEVER dataset, which was previously introduced as a standard benchmark for claim verification using Wikipedia-based evidence. A summary of the performance is shown below, while additional plots can be found in the Appendix.

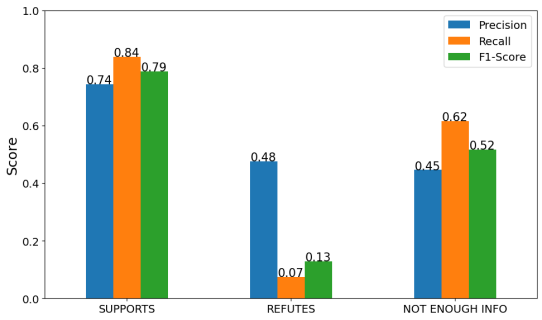

Figure 4.11: Evaluation metrics for FEVER before fine-tuning.

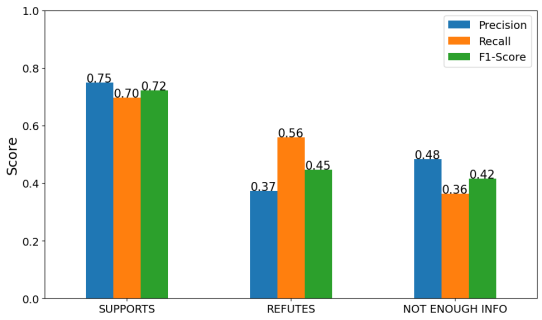

Figure 4.12: Evaluation metrics for FEVER after fine-tuning.

Figures 4.11 and 4.12 show the evaluation metrics before and after fine-tuning. Initially, the model is strongly biased towards `SUPPORTS` and `NOT ENOUGH INFO`, which led to very poor results for the `REFUTES` class. Fine-tuning significantly improves the `REFUTES` predictions, increasing the F1-score from 0.13 to 0.45, and the recall from 0.07 to 0.56. The performance on the `SUPPORTS` class remains stable with a slight decrease in recall from 0.84 to 0.70 and a slight decrease in F1-score from 0.79 to 0.72. Similarly, the performance of the `NOT ENOUGH INFO` class decreases after fine-tuning, with the most noticeable drop being the recall, which decreases from 0.62 to 0.36.

It is important to note that the developed fact-checking pipeline does not use evidence from Wikipedia. This mismatch can lead to errors, especially if claims in the dataset are outdated or rely on information that is not accessible through the

given evidence retrieval mechanisms. While FEVER is not an ideal benchmark to test the pipeline, it still provides valuable insight into how fine-tuning affects the model's performance.

4.4.3 SciFact

In addition to the FEVER dataset, the model is also evaluated on the SciFact dataset. As explained previously, it focuses on scientific claims taken from research abstracts. Additional plots can be found in the Appendix.

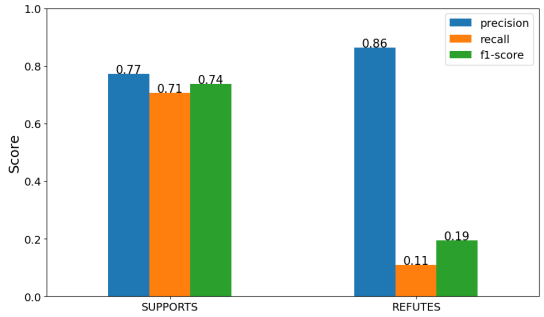

Figure 4.13: Evaluation metrics for SciFact before fine-tuning.

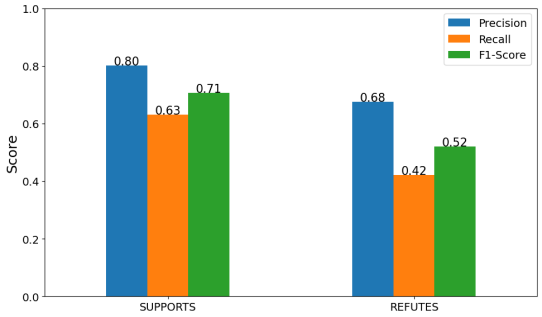

Figure 4.14: Evaluation metrics for SciFact after fine-tuning.

Figures 4.13 and 4.14 show the evaluation metrics before and after fine-tuning. Initially, the model achieved a strong performance on the `SUPPORTS` class, but performed poorly on the `REFUTES` class. Fine-tuning significantly improves the `REFUTES` class, increasing the recall from 0.11 to 0.42 and the F1-score from 0.19 to 0.52, while precision decreases from 0.86 to 0.68. The performance of the `SUPPORTS` class remains stable, with similar precision and F1-score, and a slight drop in recall from 0.71 to 0.63.

It is important to note that SciFact contains relatively difficult claims, as they were taken from specific scientific literature. Despite this challenge, the results show that fine-tuning significantly improves the model's performance in detecting refuted claims, even if the claims are highly technical and domain-specific.

4.5 Discussion

The empirical results show that fine-tuning the Mistral-7B model significantly improves its performance for the fact-checking task. This is especially visible in the results for the `REFUTES` class, which is the main objective of this work. Before fine-tuning, the model had a strong bias towards predicting the `SUPPORTS` class and failed to detect many instances of misinformation, especially for subtle or very subtle

claims. After fine-tuning, recall and F1-score for the `REFUTES` class increases significantly across all datasets, confirming the hypothesis that task-specific adaptation can improve the model towards more decisive and robust decisions. However, while the hypothesis is supported overall, there are still limitations that remain, especially in handling very subtle misinformation and in retrieving relevant evidence.

Furthermore, the analysis shows that not all improvements are from fine-tuning alone. The pipeline architecture introduced in this study shows where performance gains occur, and where bottlenecks are present. For example, the performance varied not only across classes but is also dependent on the type of misinformation and the source of the retrieved evidence. The system consistently struggled with very subtle misinformation. Additionally, the source-based analysis shows that the most frequently retrieved evidence, which is the organic result, performs the worst. In contrast, the sources that are retrieved the least, for instance, AI Overviews and knowledge graphs, perform the best.

When evaluated in the context of existing research, the findings align well with previously reported improvements observed for fine-tuning and parameter-efficient adaptation, which have shown improvements in classification accuracy and related metrics without the need for full retraining. Finally, the evaluation strategy used in this work proved particularly valuable to analyze the system's behavior and identify areas to improve. By combining synthetic and real-world datasets, and analyzing the outputs across multiple areas (class, misinformation type, source), this work offers evidence where future improvements should be made.

Chapter 5

Conclusion, Limitations and Future Work

5.1 Conclusion

The primary aim of this study is to investigate how effective a fast, modular fact-checking pipeline can be when combining Google Search-based evidence retrieval with a fine-tuned LLM, specifically Mistral-7B. This goal is divided into two main objectives: first, to design and implement a robust pipeline capable of verifying claims automatically and efficiently, using real-time web data; and second, to improve the classification accuracy of the system by fine-tuning the language model on synthetic task-specific data.

To achieve these goals, a system is developed that consists of four stages: sentence splitting, question generation, evidence retrieval, and claim-evidence classification. For classification, Mistral-7B is used in two variants: as an instruction-tuned version and as a custom fine-tuned version, which is trained on a self-created synthetic dataset, containing claim-evidence pairs with labels. The synthetic dataset is specifically designed to simulate the evidence snippets retrieved by the SerpAPI in the evidence retrieval stage.

The empirical evaluation covers three datasets: GPT2500, FEVER, and SciFact, where GPT2500 is used as the representative dataset for in-depth analysis. Across all evaluation metrics, the results confirm the effectiveness of fine-tuning Mistral-7B to the domain-specific task. Most importantly, the recall and F1-score for the `REFUTES` class improves substantially after fine-tuning, addressing one of the key weaknesses of the instruction-tuned model, which initially struggled to identify refuted claims and is often biased towards the `SUPPORTS` class. Further breakdowns by misinformation type and snippet source show important structural insights. The

model performs much better on obvious false claims and verifiable truths, but struggles with subtle misinformation. Additionally, a key observation is that the most frequently retrieved evidence sources perform the worst, whereas more structured sources such as AI Overviews or knowledge graphs, even though they are less common, are far more effective.

When evaluating the research question, **How effective is a fast fact-checking pipeline combining Google Search-based evidence retrieval and a fine-tuned LLM (Mistral-7B) for claim verification?**, the empirical evaluation of the system indicates that this approach technically works, is relatively efficient, and presents competitive classification results, especially after fine-tuning. The system is capable of verifying a wide range of claims using real-time web content. It offers a dynamic alternative to traditional fact-checking pipelines that rely on static sources. While the results are promising, certain limitations remain, especially in evidence retrieval and in handling subtle misinformation, which are discussed in the following section.

In conclusion, this study contributes a working prototype of a real-time, LLM-based fact-checking pipeline. It demonstrates the strengths of combining web search with transformer models, highlights the measurable benefits of task-specific fine-tuning, and provides a detailed analysis of where the limitations remain. These insights lay the groundwork for future research and could help guide the development of more reliable and practical fact-checking systems.

5.2 Limitations

While the developed fact-checking pipeline shows promising results, especially after fine-tuning the validation model, several limitations at both the system level and the overall project level remain.

Limitations of System Components

One key limitation is the evidence retrieval step. The system uses Google's SERP, which presents challenges in both accessibility and evidence quality. The SerpAPI has strict rate limits as well as usage costs, which limits the number of requests and makes it difficult to scale or experiment with different retrieval strategies. Furthermore, many of the retrieved snippets, especially from the organic search results, are not complete sentences, or are very vague and unreliable as evidence.

A second challenge is the question generation step, where claims are converted into questions to improve the evidence retrieval. The issue is that the quality of the

generated questions is not systematically evaluated. As a result, conversion errors, such as misinterpreting the original claim or introducing unintended ambiguity, can lead to irrelevant or low-quality search results.

Furthermore, the FEVER dataset poses a limitation due to the origin of the evidence. It exclusively relies on Wikipedia and does not reflect the inconsistency of real-world web data. Therefore, the performance on the FEVER dataset can be misleading due to the resulting evidence mismatch.

Project-Level Limitations

From a broader perspective, the pipeline assumes that the input is already a clearly formulated claim. It does not perform claim detection, which is an important step in many real-world fact-checking applications. Without this, the system is limited in an open text setting.

Another important goal of this work is to improve the detection of refuted claims. While fine-tuning significantly increases recall and F1-scores, the model still struggles with subtle misinformation. The system also performs poorly on political statements or quotes, such as "Person X said Y". The problem lies in how the evidence is retrieved. Many statements by people, especially when made recently, will not be found with the current evidence retrieval setup. These limitations lead to a number of possible improvements, which are discussed in the following section.

5.3 Future Work

While this study shows that a fact-checking pipeline using Google search-based evidence retrieval and a fine-tuned language model has great potential, it also highlights areas where future work is needed. The main research question, whether such a system can effectively verify claims, has been answered positively, although there are some limitations remaining.

First, the quality of the retrieved evidence remains the most important bottleneck with regards to improving the verification accuracy. Replacing the current retrieval mechanism with a more advanced system could significantly improve the performance of the pipeline. This could mean integrating document rerankers, source credibility scoring, or dedicated retrieval models that are optimized for factual consistency. Furthermore, the question generation step, which transforms claims into questions, could be improved. Since the question generation model is not evaluated in isolation, future work should consider a more controlled question generation strategy or alternative prompting approaches. Another important limitation is the

lack of claim detection. Currently, the pipeline can only process claims as input. For real-world applications such as automated news or social media fact-checking, future systems need to include a claim-filtering mechanism that can automatically extract relevant claims. Furthermore, while the pipeline shows significant improvements in detecting refuted claims, the performance on subtle and very subtle misinformation remains relatively weak. Addressing this will improve the overall performance of the developed system.

In conclusion, while the pipeline presents a strong baseline and shows that fine-tuning transformer models can significantly improve claim classification, further work is needed to improve the retrieval precision, question generation, claim identification, and detection of subtle misinformation. Each of these areas shows a clear path for advancing the system towards a deployment in real-world fact-checking scenarios.

Appendix A

Evaluation Results for the FEVER and the SciFact datasets

A.0.1 FEVER

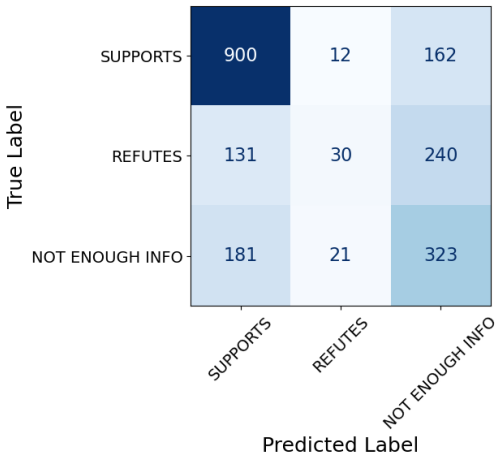

Figure A.1: Confusion matrix for FEVER before fine-tuning.

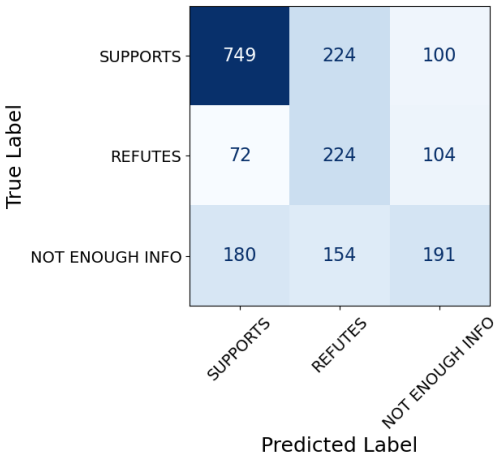

Figure A.2: Confusion matrix for FEVER after fine-tuning.

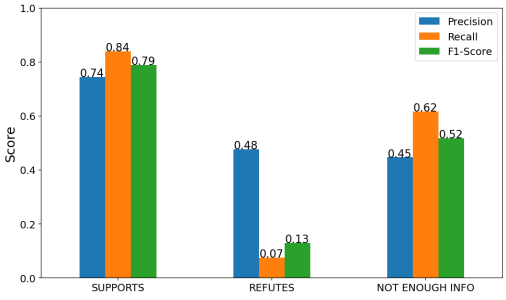

Figure A.3: Evaluation metrics for FEVER before fine-tuning.

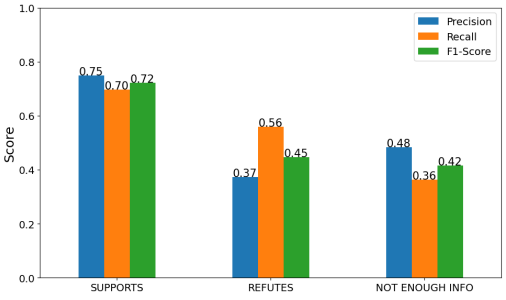

Figure A.4: Evaluation metrics for FEVER after fine-tuning.

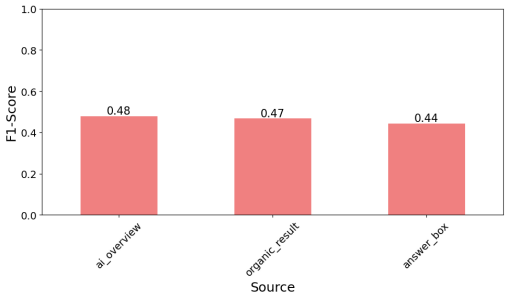

Figure A.5: F1-score for FEVER by source before fine-tuning.

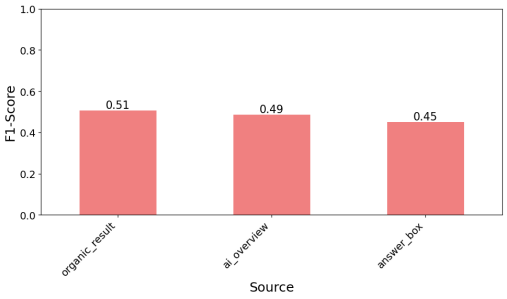

Figure A.6: F1-score for FEVER by source after fine-tuning.

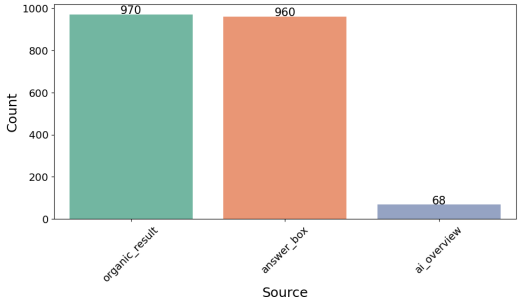

Figure A.7: Snippet count for FEVER by source.

A.0.2 SciFact

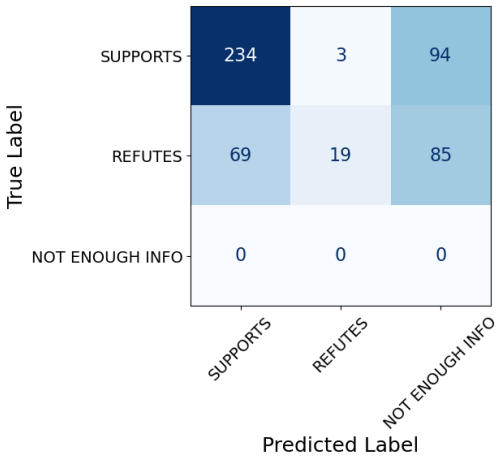

Figure A.8: Confusion matrix for SciFact before fine-tuning.

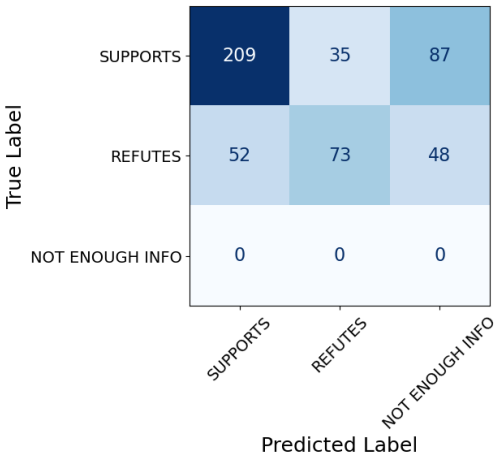

Figure A.9: Confusion matrix for SciFact after fine-tuning.

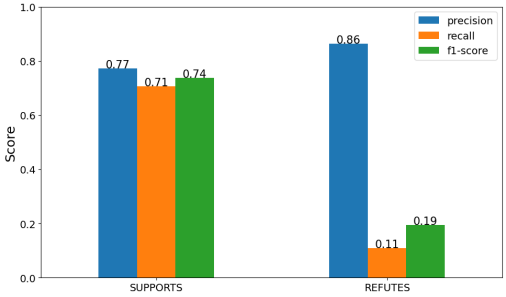

Figure A.10: Evaluation metrics for SciFact before fine-tuning.

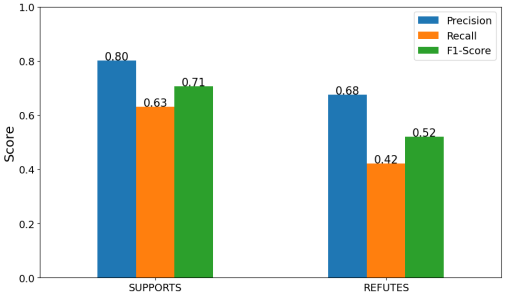

Figure A.11: Evaluation for SciFact after fine-tuning.

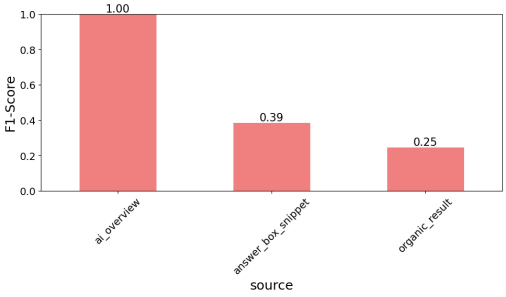

Figure A.12: F1-score for SciFact by source before fine-tuning.

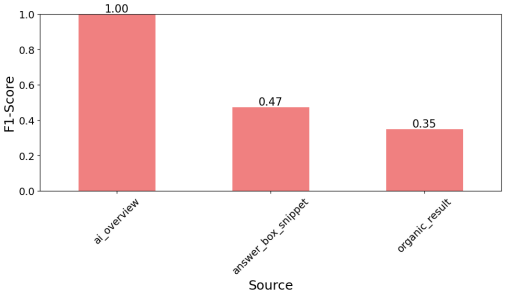

Figure A.13: F1-score for SciFact by source after fine-tuning.

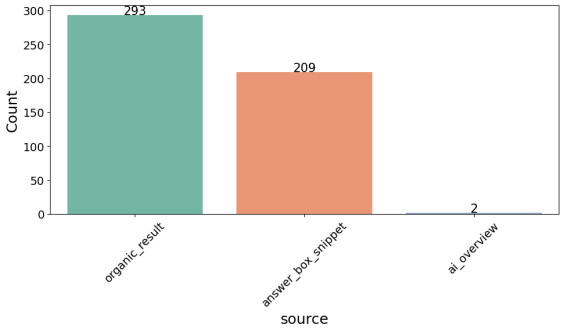

Figure A.14: Snippet count for SciFact by source.

Appendix B

Datasets

The datasets used to evaluate the performance of the pipeline, including retrieved evidence and model outputs for both the base and the fine-tuned Mistral-7B model, are available at:
https://github.com/doeniz/fact-verification-datasets-with-evidence/tree/main/datasets

Bibliography

[1] Daniel Jurafsky and James H Martin. *Speech and Language Processing*. Pearson, 2023.

[2] Benjamin Riedel, Isabelle Augenstein, Georgios P. Spithourakis, and Sebastian Riedel. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *CoRR*, abs/1707.03264, 2017.

[3] Xia Zeng, Amani S. Abumansour, and Arkaitz Zubiaga. Automated fact-checking: A survey. *Lang. Linguistics Compass*, 15(10), 2021.

[4] Will Godel, Zachary Sanderson, Kevin Aslett, Jonathan Nagler, Richard Bonneau, Nathaniel Persily, and Joshua A. Tucker. Moderating with the mob: Evaluating the efficacy of real-time crowdsourced fact-checking. *Journal of Online Trust and Safety*, 1(1), oct 2021.

[5] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[6] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Trans. Mach. Learn. Res.*, 2024, 2024.

[7] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. A survey on text classification: From shallow to deep learning. *CoRR*, abs/2008.00364, 2020.

[8] Marco Loog. Supervised classification: Quite a brief overview. *CoRR*, abs/1710.09230, 2017.

[9] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and verification. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the*

2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers), pages 809–819. Association for Computational Linguistics, 2018.

[10] David Wadden, Kyle Lo, Bailey Kuehl, Arman Cohan, Iz Beltagy, Lucy Lu Wang, and Hannaneh Hajishirzi. Scifact-open: Towards open-domain scientific claim verification. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 4719–4734. Association for Computational Linguistics, 2022.

[11] Mars Gokturk Buchholz. Assessing the effectiveness of GPT-3 in detecting false political statements: A case study on the LIAR dataset. *CoRR*, abs/2306.08190, 2023.

[12] Maram Hasanain and Tamer Elsayed. Studying effectiveness of web search for fact checking. *J. Assoc. Inf. Sci. Technol.*, 73(5):738–751, 2022.

[13] Juraj Vladika and Florian Matthes. Comparing knowledge sources for open-domain scientific claim verification. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024 - Volume 1: Long Papers, St. Julian's, Malta, March 17-22, 2024*, pages 2103–2114. Association for Computational Linguistics, 2024.

[14] Advanced Web Ranking. Ai overview study for 8,000 keywords in google search, 2024. Accessed: 09.07.2025.

[15] Rich Sanger. Google ai overviews: The role of large language models and google gemini, 2024. Accessed: 09.07.2025.

[16] Max Delaney. Google's ai overviews are often so confidently wrong that i've lost all trust in them, 2025. Accessed: 09.07.2025.

[17] Google. How google's knowledge graph works, 2025. Accessed: 09.07.2025.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural*

Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008, 2017.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[20] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[21] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023.

[22] Mistral AI. Announcing mistral 7b, 2023. Accessed: 20.06.2025.

[23] Aditya Venkatesh. Mistral 7b explained — the open-weight llm that beats llama 2, 2023. Accessed: 20.06.2025.

[24] Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright

Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll L. Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, and Dane Sherburn. Gpt-4o system card. *CoRR*, abs/2410.21276, 2024.

[25] Guanwen Xie, Jingzehua Xu, Yiyuan Yang, Yimian Ding, and Shuai Zhang. Large language models as efficient reward function searchers for custom-environment multi-objective reinforcement learning. *CoRR*, abs/2409.02428, 2024.

[26] Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. Chatgpt outperforms crowd-workers for text-annotation tasks. *CoRR*, abs/2303.15056, 2023.

[27] Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Iqbal khan, and Arsalan Shahid. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *CoRR*, abs/2408.13296, 2024.

[28] Luping Wang, Sheng Chen, Linnan Jiang, Shu Pan, Runze Cai, Sen Yang, and Fei Yang. Parameter-efficient fine-tuning in large language models: a survey of methodologies. *Artif. Intell. Rev.*, 58(8):227, 2025.

[29] Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *CoRR*, abs/2303.15647, 2023.

[30] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7534–7550. Association for Computational Linguistics, 2020.

[31] Explosion AI. Spacy usage: Linguistic features - sentence boundary detection, 2025. Accessed: 27.04.2025.

[32] Sorouralsadat Fatemi, Yuheng Hu, and Maryam Mousavi. A comparative analysis of instruction fine-tuning large language models for financial text classification. *ACM Trans. Manag. Inf. Syst.*, 16(1):1–30, 2025.

[33] LLC SerpApi. Serpapi: Advanced features, 2025. Accessed: 29.04.2025.

[34] OpenAI. Openai platform documentation: Gpt-4o, 2025. Accessed: 31.07.2025.

[35] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021.

[36] Hugging Face. Transformers documentation, 2025. Accessed: 31.07.2025.

[37] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *CoRR*, abs/2312.12148, 2023.

Erklärung

gemäss Art. 30 RSL Phil.-nat.18

Name/Vorname: Demir Deniz Rino

Matrikelnummer: 19-918-978

Studiengang: Informatik

Bachelor ☑ Master ☐ Dissertation ☐

Titel der Arbeit: Towards Reliable Fact-Verification Systems

LeiterIn der Arbeit: PD Dr. Kaspar Riesen

Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe r des Gesetzes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.
Für die Zwecke der Begutachtung und der Überprüfung der Einhaltung der Selbständigkeitserklärung bzw. der Reglemente betreffend Plagiate erteile ich der Universität Bern das Recht, die dazu erforderlichen Personendaten zu bearbeiten und Nutzungshandlungen vorzunehmen, insbesondere die schriftliche Arbeit zu vervielfältigen und dauerhaft in einer Datenbank zu speichern sowie diese zur Überprüfung von Arbeiten Dritter zu verwenden oder hierzu zur Verfügung zu stellen.

Ort/Datum Murten, 10.08.2025

Unterschrift D. Demir