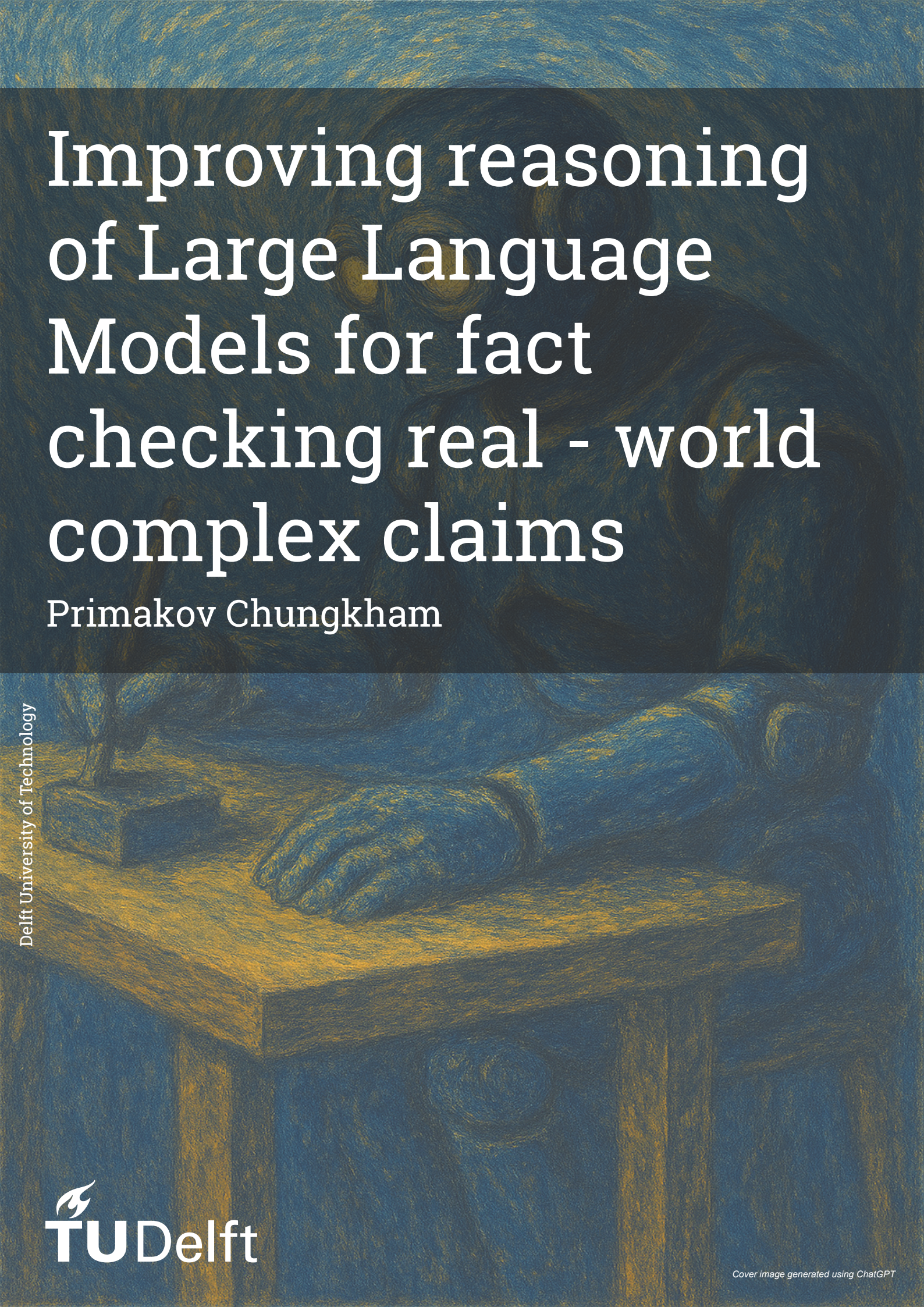
Improving reasoning of Large Language Models for fact checking real - world complex claims

Primakov Chungkham

Delft University of Technology



TU Delft

Cover image generated using ChatGPT

Improving reasoning of Large Language Models for fact checking real - world complex claims

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Primakov Chungkham
born in Imphal, India

Committee Chair:	Dr. Avishek Anand
First Core Member:	Dr. Kubilay Atasu
Daily Supervisor:	Dr. Venktesh Viswanathan
Project Duration:	September 2024 – May 2025
Faculty:	Faculty of EEMCS, TU Delft

Web Information Systems Group
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Preface

This thesis marks the culmination of my Master's journey in Computer Science at TU Delft. It represents not only a technical undertaking but also a personal and intellectual exploration into how large language models (LLMs) can be used to tackle the growing challenge of fact-checking real-world claims. In conducting this research, I aimed to bridge theoretical developments in natural language processing with their practical application in ensuring information integrity. Throughout this process, I learned how to design and carry out structured, independent research - formulating hypotheses, evaluating results, and iterating on insights.

I owe immense gratitude to my supervisors, Dr. Venktesh Viswanathan and Dr. Avishek Anand. Their steady guidance, technical insight, and patience were instrumental throughout this thesis. They challenged me to think critically and helped sharpen the direction of my work, all while creating an environment that allowed for creativity and independence. I am also grateful to the Web Information Systems group for the knowledge exchange sections, and their valuable feedback on my work.

I am grateful to TU Delft for the providing resources, guidance, and an environment to execute free thinking, ask the right questions and conduct impactful research. The infrastructure and support I received here laid the foundation for this project and made it possible to pursue my interests at the frontier of AI research.

Most importantly, I want to express my profound gratitude to my mother. Her unwavering support and endless sacrifices made everything in this journey possible. I am also thankful to my family and friends for their constant encouragement and emotional support, which helped me stay grounded and motivated.

Lastly, a small note of appreciation to my ever-reliable laptop, which stood by me through countless late nights, long experiments, and endless lines of code. Its quiet resilience made this thesis possible in more ways than one.

Primakov Chungkham
Delft, May 2025

Abstract

Improving reasoning of Large Language Models for fact checking real - world complex claims

Abstract

The growing volume of online misinformation has increased the demand for automated fact-checking systems. While large language models (LLMs) have demonstrated potential in this domain, real-world claims are complex that challenge standard prompting techniques. These claims are *multi-aspect*, requiring integration of evidence across different sources. Among these, a challenging subset are claims with *Conflicting* labels, where different parts of the claim support opposing veracity labels. Both cases demand nuanced, context-sensitive reasoning that LLMs struggle to perform reliably. This thesis investigates two methods to enhance LLM reasoning for fact-checking - *claim decomposition* and *test-time scaling*. Claim decomposition breaks down complex claims into simpler sub-questions, promoting more structured reasoning. While this improves performance on *Conflicting* claims, it can degrade accuracy for straightforward claims, particularly those labeled as *True*. To mitigate this, an adaptive decomposition strategy is proposed, selectively applying decomposition only when beneficial. A taxonomy of reasoning failure - termed *overthinking* - is identified, where the model becomes unnecessarily strict due to noisy evidence or overly specific sub-questions. To further address this issue, *test-time scaling* using a reward model is employed to rank candidate outputs. This approach yields an 18.8% relative improvement in macro F1-score over the baseline and reduces overthinking by encouraging context-aware leniency. Together, these findings underscore the importance of targeted reasoning strategies for improving the robustness and reliability of LLM-based fact-checking.

Contents

List of Tables

List of Figures

1 Introduction

1.1. Background and Motivation

1.1.1. Challenges of fact-checking real-world complex claims

The proliferation of misinformation across digital platforms has made verifying factual accuracy at scale more critical than ever [1, 2]. From public health to political discourse, false claims can spread rapidly and influence real-world outcomes. Many real-world claims contain a numerical or quantitative figure in them [3]. Such claims are more likely to be perceived as correct due to their associated numerical figures - an effect termed as *numeric-truth-effect* [4]. This effect can have serious real-world consequences. A prominent illustration is the U.S. opioid crisis, where Purdue Pharma used misleading numerical claims—such as falsely asserting that the addiction risk of OxyContin was less than 1%—to market the drug as safe [5]. These precise figures, though inaccurate, contributed to the widespread perception of safety and ultimately fueled large-scale overprescription and addiction.

In response to the need for scalable fact-checking, automated fact-checking has emerged as a promising computational approach to assess the truthfulness of claims with minimal human intervention. This process typically involves evaluating a claim against supporting evidence to determine whether it is corroborated or contradicted, generating an explanation for the decision. The stage of this process known as *veracity prediction* is responsible for making this determination, outputting a decision in the form of a *veracity label* (e.g., *Supported*, *Refuted*) along with a corresponding textual explanation, known as the *justification* [6, 7]. Recent advancements in large language models (LLMs) have enabled researchers to frame veracity prediction as a summarization task [8, 9], allowing systems to generate both a classification and its corresponding natural language explanation.

Despite these advances, verifying real-world claims remains a challenging task that demands intense reasoning capabilities. A key difficulty lies in the inherent complex nature of such claims, which are often multi-faceted and require the verification of several interrelated aspects to determine their overall truthfulness. Furthermore, the supporting evidence for these individual aspects is frequently distributed across multiple sources, rather than contained within a single document [10, 11, 12, 13]. Consequently, a language model must be capable of iden-

tifying the various aspects of a claim, retrieving the appropriate pieces of evidence for each, evaluating the alignment between the claim and the evidence, and aggregating these individual verifications to produce a final veracity label.

Figure 1.1 illustrates the complexity of real-world claims through a representative example. The claim in question asserts that the Obama administration offered $900 million to Hamas - a designated terrorist organization - for the reconstruction of Gaza. Upon closer examination, the evidence reveals that while the monetary amount ($900 million) is accurate, the aid was directed to *Gaza* rather than to *Hamas*. Additionally, a separate source confirms that *Hamas* is officially recognized as a terrorist organization. Relating these pieces of evidence leads to the correct veracity label of *False* for the claim. This example underscores the need for comprehensive reasoning that considers all relevant aspects of a claim, ensuring that the final veracity decision reflects the truthfulness of each component.

[Claim]: The Obama administration offered "$900 million to Hamas, a recognized terrorist organization, to rebuild Gaza."

[Evidences]:

1. .... washington the obama administration intends to provide some $900 million to help rebuild gaza after the israeli incursion that ended last month, administration officials said monday. by seeking to aid gazans but not hamas, the .....
2. hamas is an islamist extremist palestinian organization that calls for the eradication of the state of israel.....

[Label]: False

Figure 1.1: An example of a complex real world claim and its multi aspect evidence.

An additional challenge in veracity prediction stems from the need to reason about claims that do not fall neatly into binary categories such as *True* or *False*. Instead, real-world fact-checking often requires solving a multi-class classification problem. Datasets like QUANTEMP [3] standardize annotations across sources into three labels: *True*, *False*, and *Conflicting*. This setup introduces greater nuance but also increases the complexity of the classification task.

Multi-class classification inherently increases the modeling challenge, as the decision boundary between classes is no longer binary and often not well-separated. This complexity is compounded in tasks where the classes are not strictly disjoint. In case of fact-checking, the *Conflicting* class exhibits significant semantic and evidential overlap with both *True* and *False*, blurring the boundaries that would otherwise guide the model's decisions. Unlike *True* or *False*, which typically involve more direct correspondence or contradiction between the claim and evidence, *Conflicting* comprises claims that blend accurate information with misleading or decontextualized elements.

A concrete example is shown in Figure 1.3. The claim that "Obama shut down the government

for 16 days to force Obamacare" contains some factual information—the government *was* shut down for 16 days in 2013, and Obamacare was central to the political conflict. However, the evidence contradicts the claim's *attribution of blame*, showing instead that it was a Republican-led Congress that attempted to defund the Affordable Care Act. This blend of partial truth and misattribution exemplifies the semantic ambiguity inherent to *Conflicting* claims.

As shown in Figure 1.2, the *Conflicting* label in QUANTEMP is mapped from a wide range of original annotations, including *Partly True*, *Mixture*, *Missing Context*, or *Miscaptioned*. These diverse interpretations lead to heterogeneous supervision signals, making it difficult for models to learn consistent decision boundaries.

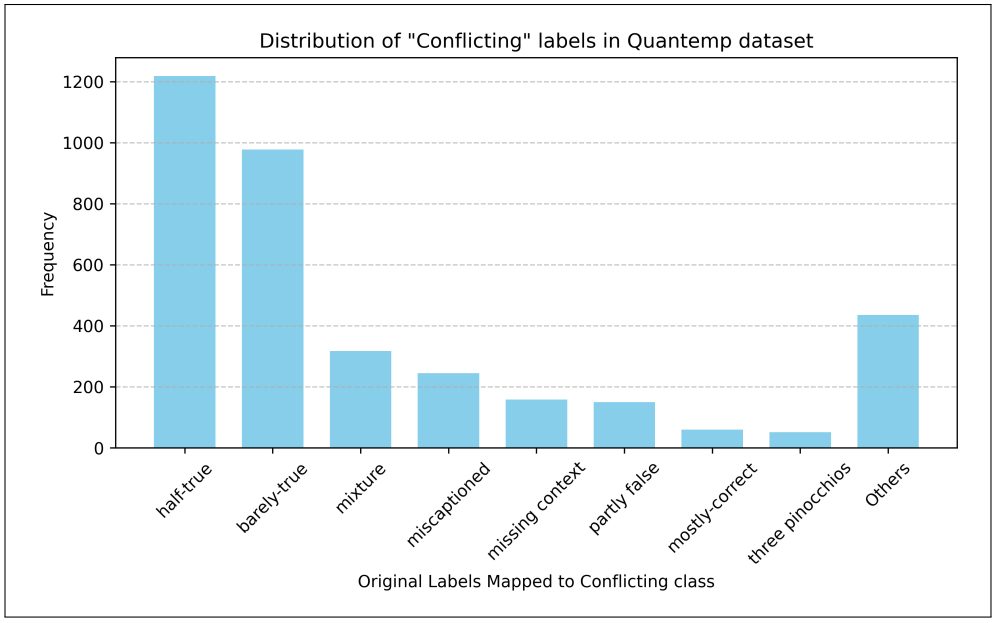


Figure 1.2: Distribution of original labels mapped to Conflicting class in Quantemp [3] dataset.

We find empirically that this class is also the hardest to predict correctly: as shown in Table 5.1, the F1 score for *Conflicting* is substantially lower than for *True* and *False* when run in *without decomposition* setting. This indicates that language models struggle to generalize over the fuzzy reasoning patterns required to identify ambiguity, conflicting context, or blended actuality. These observations highlight a key modeling bottleneck in real-world fact-checking: language models are not inherently equipped to reason about subtle contradictions or misattributions, especially when these are embedded in otherwise factually plausible statements.

1.1.2. Decomposition to address the multi-aspect and ambiguity challenges of real-world claims

Owing to the multi-aspect nature and semantic ambiguity inherent in many real-world claims, performance of language models in veracity prediction remains suboptimal [3, 13, 12]. This issue is particularly acute for claims labeled as *Conflicting*, where elements of truth are interwoven with misleading or exaggerated content. As shown in Figure 1.3, such claims require

[Claim]: "October 2013, Obama shut down the government for 16 days to force Obamacare.

[Evidences]: .... in 2013, when the government was partially closed for 16 days after another republican-led congress tried to use budget negotiations to defund barack obamas signature affordable care act, widely known as obamacare ....

[Label]: Conflicting

Figure 1.3: An example of a "Conflicting" claim where certain aspect of the claim is True while others are exaggerated.

methods to reason over subtle contradictions and contextual nuance capabilities that current language models often lack. Additionally, these models struggle with accurately identifying and processing numerical data [14], which further impedes their ability to assess claims involving numerical figures.

To address these limitations, several researchers have proposed decomposing claims into sub-claims or sub-questions based on various properties of the claim [13, 12, 15, 11]. Using these sub-questions as a framework for evidence retrieval and reasoning has been shown to enhance model performance [16]. For example, the multiple aspects of the claim depicted in Figure 1.1 can be broken down into the following sub-questions:

1. Did the Obama administration offer $900 million to Hamas?
2. Was the money offered to rebuild Gaza?
3. Is Hamas a recognized terrorist organization?

Reasoning using these aspects enables the veracity prediction model to identify relevant evidence (highlighted in red in Figure 1.1), guiding the overall veracity prediction process.

1.1.3. Shortcoming of decomposition

While decomposition strategies have been shown to enhance the reasoning capabilities of language models, they also introduce certain limitations. A prominent issue is over-decomposition, where decomposing claims into excessively fine-grained sub-questions can impair the performance of veracity predictions [17]. Qualitative analyses of preliminary results suggest that this issue is closely linked to a well-documented artifact in large language models (LLMs) known as *overthinking*. This refers to a tendency of LLMs to deviate from correct reasoning paths by overemphasizing superficial keyword cues in the input [18, 19]. As shown in Table 5.1, this phenomenon is particularly detrimental for claims labeled as *True*, where decomposition reduces prediction accuracy. Supporting this observation, Table 5.2 highlights that overthinking accounts for the majority of decomposition-related performance drops in *True* claims.

In contrast, applying decomposition selectively-specifically to claims labeled as *Conflicting* yields a notable improvement in performance across all veracity classes, as illustrated in Table 5.1. This result is grounded in the observation that *Conflicting* claims pose greater reasoning challenges due to their inherently complex and often ambiguous nature, as discussed in Section 1.1.1. These findings underscore the limitations of uniform decomposition strategies and motivate the design of an *adaptive decomposition* mechanism that tailors decomposition decisions based on claim-specific properties. In this study, we explore principled and robust approaches for guiding decomposition selectively, aiming to improve predictive performance.

1.1.4. Shortcoming of relying on a top 1 output sequence of LLMs

While adaptive decomposition has been shown to enhance the performance of veracity prediction tasks, it does not directly address reasoning issues like *overthinking* and *numerical complexity* that exist in the output sequence generated by the language models. Prior work has demonstrated that the top-1 output of large language models is often unreliable for tasks requiring complex reasoning [20, 21]. This insight has motivated the development of techniques that involve sampling multiple output sequences per input and aggregating these to arrive at a more robust final prediction [21, 22]. These approaches, which require increased computational resources during inference, fall under the umbrella of test-time scaling [23].

A particularly effective strategy within this paradigm involves training a reward model to select the most promising output sequence from a set of candidates [24, 25, 26]. When applied to reasoning-intensive tasks, such methods have not only demonstrated improvements in task performance but have also enhanced the correctness of the underlying reasoning process [26].

It is hypothesized that sampling a sufficient number of independent decoding paths from the model, given an input comprising a claim, supporting evidence, and optional sub-questions, yields a high likelihood of generating at least one sequence that exhibits valid reasoning and leads to a correct verdict. Empirical analysis provides support for this hypothesis: when 10 decoding paths are sampled per claim, it is observed that a substantial proportion of claims are correctly reasoned in at least one sequence (see upper bound results in Table 5.4). Consequently, the objective is to train a reward model that can identify and select the correctly reasoned sequence, thereby improving the performance of veracity prediction. Furthermore, an investigation is undertaken to determine whether the reward model genuinely learns to recognize higher-quality reasoning, rather than solely optimizing performance.

1.2. Research Questions

To address the discussed shortcomings in the reasoning capabilities of LLMs, we pose the following research questions:

1. **RQ1:** How does claim decomposition affect veracity prediction in fact checking complex claims?

2. **RQ2:** Can adaptive decomposition of the claims help improve performance of veracity prediction?
3. **RQ3:** How does Reward Model (RM) guided test-time scaling improve reasoning and performance of veracity prediction?

1.3. Scientific Contribution

The contributions of this work, derived from addressing the proposed research questions, are as follows:

1. **Comprehensive evaluation of claim decomposition strategy:** An in-depth study is conducted on the effect of claim decomposition on LLM-based veracity prediction performance. Furthermore, qualitative analysis identifies and categorizes the taxonomies of errors where decomposition is not necessary.
2. **A principled approach to adaptive decomposition for efficiency and robustness:** Two methods are proposed for adaptive decomposition, enabling decomposition decision to be made based on the structural complexity of the claims. This approach increase veracity prediction performance while also reducing computational overhead by avoiding decomposition for certain claims when unnecessary making it more time and cost efficient (see Appendix A.6 for details).
3. **Reward modeling for reasoning-aware sequence selection:** It is shown that training a reward model to evaluate and select from multiple candidate outputs leads to improvement in veracity prediction performance. Qualitative analysis also indicate that the reward model preferentially selects outputs that exhibit more coherent and logically sound reasoning.

1.4. Thesis Outline

The report is laid out in the following way. Chapter 2 discusses current related work pertinent to the research objectives of this study. Chapter 3 dives into the methodology used to perform adaptive decomposition, and training and inference of a reward model to select the best reasoning path. Chapter 4 discusses the experiments conducted and the implementation specification used to answer the research questions. Chapter 5 discusses the results obtained and relate them with the research questions and existing literature. Finally, Chapter 6 concludes the thesis while discussing limitations and directions for future improvements.

2

Related Work

2.1. Real world claims dataset for automated fact checking

After the formal definition of automated fact checking [7], there have been many attempts to design a dataset for training and evaluating models on this task. Among them, FEVER [27] is a prominent dataset that consists of claims from Wikipedia rephrased by crowd workers. Such claims are called synthetic claims which are artificially created from a well-known fact and they do not accurately represent the real-world claims fact-checked by the journalists [11]. Following up on this, there have been several synthetic claims datasets sourced either from Wikipedia [28, 29, 30] or from domain-specific sources [31, 32, 33].

On the contrary, real-world claims datasets are created by consolidating actual claims from the web or fact-checking organizations. These claims are obtained from various fact-checking websites. First attempt in creating a real-world dataset [7] collected claims from two fact-checking websites - (i) Channel 4[1] and Politifact[2]. The different veracity labels are aligned into a five-point scale: *TRUE*, *MOSTLYTRUE*, *HALFTRUE*, *MOSTLYFALSE*, and *FALSE*. This resulted in a dataset consisting of 221 claims and their corresponding justifications. Inspired by this, LIAR [34] dataset was introduced with around 12,800 political claims with six veracity labels. These claims are accompanied by claims metadata such as their identity and context. Owing to its large sample size, it is more suitable for training and testing machine learning models.

However, these datasets lack the evidence to predict the veracity of the claims. To address this gap, LIAR-PLUS [35] extracted the justification documents for the claims present in the LIAR dataset [34] and used them as evidence alongside the claim to perform the claim veracity prediction task. Results indicate that using this *evidence*, improves veracity prediction performance. PolitiHop [10] dataset modeled fact-checking task along the line of multi-hop reasoning [36, 37]. It annotated multiple pieces of connected evidence required to predict the veracity of a claim, thus acknowledging the complex nature of real-world claims. There are several datasets with real-world claims that deal with different domains such as health [38, 8], climate [39], multi-lingual [40] and open domain [41, 42]. Recent attempts in creating a

[1] http://blogs.channel4.com/factcheck/

[2] http://www.politifact.com/ truth-o-meter/statements/

real-world fact-checking dataset have modeled it as a decomposition task with works such as QABriefs [12], CLAIMDECOMP [13], AVERITEC [11], and QUANTEMP [3]. These datasets often decompose the claims into sub-claims or sub-questions which inherently assume that they are complex in nature. These decomposition techniques are discussed in detail in section 2.2.

2.2. Claim Decomposition in Fact Checking

The first attempt to decompose a fact-checking claim into sub-claims/sub-questions was the creation of the QABriefs dataset [12] which consists of question-answer pairs pertaining to a claim, generated by crowd workers. These questions are generated around *briefs* - a concept inspired by briefing someone on a situation. A BART model [43] was fine-tuned to automatically generate questions and another was trained to generate answers. The results indicate that using QABrief increases the fact-checking accuracy while reducing the fact-checking time thus making the task more efficient. Inspired by this, CLAIMDECOMP [13] created a human-annotated dataset by decomposing complex political claims into a set of yes-no questions. This set of questions should collectively cover the entire aspect of the claim and each question is relevant in determining the veracity of the claim. Experimental results indicate that evidence retrieved using these generated questions outperforms veracity prediction performance when using only the claim, hence showing the effectiveness of these questions in claim-veracity prediction.

Varifocal [44] proposed to generate questions around *focal points* of a claim. Focal points are either the spans from the claims which are extracted from its syntactic parse tree or metadata such as the source of the claim, the speaker's name, or the date of the claim. They evaluated the quality of the generated questions using both automatic metrics and human evaluation. Human evaluators evaluated based on intelligibility, clarity, relevance, and informativeness. Results indicate that their method generates better quality questions against QABriefs' BART model [12] and another model trained on the SQUAD dataset. However, ProgramFC [15] took a different path and proposed to decompose a claim into simpler small sub-tasks. They used the Codex model [45] in a few-shot setting to generate a series of sub-tasks/programs and call an off-the-shelf executor to execute these sub-tasks independently. The function of these sub-tasks is to verify a simpler claim, answering a question or performing simple logic reasoning. Their method outperforms other fact-checking baselines when compared on the HOVER [36] and FEVEROUS-S [46] datasets.

AVERITEC [11] collects real-world claims from 50 different organizations and employs human annotators to generate multiple question-answer pairs to determine the veracity of the claims. Unlike CLAIMDECOMP [13] which limits the questions to be yes/no type, they allow the answers to be abstractive, extractive or boolean. To determine the veracity of the claim, they fine-tune a BERT-large model [47] to label each question-answer pair as supporting, refuted or irrelevant to the question. The final verdict is determined by aggregating each question-

answer verdict to predict one of the four veracity labels - supported, refuted, not enough evidence, or conflicting evidence/cherry picking. Results indicate general difficulty in predicting the veracity labels of the Conflicting evidence class.

Similarly, QUANTEMP [3] collects real-world claims from various fact-checking organizations and identifies numerical claims which include numbers, units, approximations (eg, roughly), or trends (”increases”). This results in roughly 15,000 numerical real-world complex claims with veracity labels - *True*, *False* or *Conflicting*. These claims are decomposed by prompting gpt-3.5-turbo [3] to generate sub-questions or sub-tasks similar to methods in CLAIMDECOMP [13] and ProgramFC [15]. These sub-questions and sub-tasks are used to fine-tune a roberta-large model [48] to predict the veracity of the claims. Results indicate good performance over all veracity classes with *Conflicting* class being relatively difficult.

2.3. Adaptive Decomposition

To the best of our knowledge, there has not been any attempt to design an explicit mechanism to perform adaptive decomposition of queries (claim) in the context of fact-checking. However, research is done in the field of RAG-based question answering to optimize the compute by avoiding retrieval for simpler queries which can be answered correctly from the model's parametric memory.

One of the early attempts at Adaptive RAG was the FLARE method [49]. In this method, the confidence of the tokens is monitored during generation. Whenever the confidence falls below a certain threshold, retrieval is triggered using the low-confidence sentence to formulate the search query. Inspired by this, DRAGIN [50] extends upon the criteria to trigger retrieval beyond confidence by incorporating a token's contribution to the subsequent context (computed using attention score), and whether the current token is a stop word[4]. Results indicate that DRAGIN outperforms FLARE in numerous multi-hop question answering [37, 51], commonsense reasoning [52] and reading comprehension tasks [53]. Similarly, Self-DC [54] leverages the confidence score of the generated token to decide either to decompose a query, retrieve additional information or to generate the answer from parametric memory.

Self RAG [55] uses special tokens called ”reflection tokens” to indicate the need to call the retrieval model on-demand. A critique model is trained to create a dataset that learns to embed these tokens wherever necessary. A final generator model is trained on this annotated dataset to learn to generate these tokens wherever necessary. Adaptive RAG [56] uses queries from single-hop [57, 58, 59] and multi-hop [60, 37, 51] datasets and annotates these queries into three levels of complexities. A complexity classifier is trained upon this dataset to decide the extent of retrieval (ranging from no retrieval to multiple iterations), a query requires to obtain the correct answer. CTRLA [61] relies on the latent representation of the query to analyze the ”confidence” with which the k-th token is generated. If the confidence value falls below

[3] https://platform.openai.com/docs/models/gpt-3.5-turbo
[4] https://en.wikipedia.org/wiki/Stop_word

a certain threshold, the retriever is triggered. The idea of extracting latent representation is inspired by the superposition hypothesis [62, 63].

2.4. Test time scaling using a reward model

Test-Time Scaling is a technique to improve the reasoning performance of LLMs by allocating additional inference-time compute. Early approaches primarily focused on generating multiple outputs per query and selecting an answer via majority voting [21] or search-based methods [22, 64, 65]. Recent developments can be broadly categorized into two lines of work: (i) modifying the proposal distribution of the model's output [66, 67, 68, 69, 70, 71, 72, 73, 74, 75], and (ii) training a reward model to select the preferred answer from multiple candidates [23, 20, 76, 25, 24, 26]. Our methodology focuses on the latter, often referred to as *search against a verifier.*

Search against verifiers involves sampling multiple outputs and using a trained reward model (or verifier) to select the best final output or the reasoning path that led to it. The earliest attempts to use verifier models focused on generating better-quality stories [76], where candidate generations were ranked based on human feedback signals. Recently, there has been growing interest in applying verifier models to mathematical reasoning tasks. Generate and Rerank [77] jointly trained an encoder-decoder model [43] to generate multiple candidate solutions to math problems and rank them with a classifier head attached on top of the decoder unit. Their solution outperforms contemporary state-of-the-art methods on numerous math problem benchmarks [78, 79]. Inspired by this, the GSM8K dataset [25] was introduced which consists of 8,000 high-quality math word problems. A verifier model was trained which outperformed fine-tuned counterparts on this benchmark.

This class of verifier models, which rank entire output sequences based on outcome quality, are generally referred to as Outcome-based Reward Models (ORMs). An alternative approach, known as Process-based Reward Models (PRMs), involves training a reward model to score the quality of individual reasoning steps leading to the final answer. By providing finer-grained feedback, PRMs encourage the model to generate more accurate and reliable intermediate steps [24]. A subsequent study [26] conducted a detailed comparison and showed that PRMs outperform ORMs when using best-of-N sampling on the MATH dataset [80]. MATH-SHEPHERD [20] trained both the outcome reward model and the process reward model for the mathematical reasoning. Their novel contribution came with an automatic process to annotate the ground-truth of the reasoning steps for training the PRM, which is inspired by Monte-Carlo Tree Search [81, 82, 83, 84]. This involves unrolling a reasoning step till completion numerous times and assigning a score to it corresponding to the number of times it reaches the correct answer. Another study [23] expanded the decoding strategy over Best-of-N by incorporating two additional strategies - (i) beam search which apply best-of-N over N final answers obtained via beam search [85, 22] and (ii) lookahead search which incorporate unrolled PRM scores of each step to perform beam search.

2.5. Related work relevancy

This section discusses relevant prior work, with a focus on how existing methods relate to and motivate the approaches adopted in this thesis. Section 2.1 briefly discusses the history of automated fact checking, their datasets and motivates the complex nature of the claims fact-checked in real life. Section 2.2 discusses the concept of claim decomposition and various ways of performing decomposition. The claim decomposition method used for this study is inspired by the CLAIMDECOMP[13] paper. Then, state-of-the-art methods to perform adaptive querying in the RAG setting is discussed (section 2.3). Though not previously explored, we hypothesize that this literature can be adopted in the fact-checking domain to perform adaptive decomposition of claims. The notion of claim complexity and one method of adaptive decomposition is inspired by the adaptive RAG [56] paper and the other is inspired by CTRLA [61].

In section 2.4, developments in test-time scaling of LLMs to improve their reasoning capabilities are discussed. To the best of our knowledge, there has been no attempt to incorporate test-time scaling to improve reasoning in fact-checking, but our work is inspired by the *search against verifier* literature. For this study, the reward model is neither trained on the final output (outcome-based reward model) nor on each reasoning step independently (process-based reward model). Instead, the whole output sequence is fed to the reward model and a score is assigned to it. To avoid confusion, the terminology *reward model* is used throughout the report to refer to the verifier/reward model.

3

Methodology

3.1. Veracity prediction pipeline

The veracity prediction pipeline with adaptive decomposition is presented in Figure 3.1. Given a claim, c, the first step is to determine whether it requires decomposition and up to what extent. Accordingly, the claim is decomposed into a set of sub-questions, $Q =< q1, q2, .., qn >$, which along with the claim is used as a query to retrieve and re-rank the top-*k* evidence, E. The sub-question set can be empty if the adaptive decomposer decides that the claim does not require decomposition. In that case, only the claim is used to retrieve and re-rank the evidence. Finally, the claim, sub-questions, and the evidence are fed to the inference model, M to predict the veracity label, $\hat{y}$ and generate justification, J. The inference model is a pre-trained language model that exhibits language understanding, generation, and reasoning capabilities. Formally, the fact-checking pipeline can be represented as:

$$(\hat{y}, J) = M(c, Q, E) \tag{3.1}$$

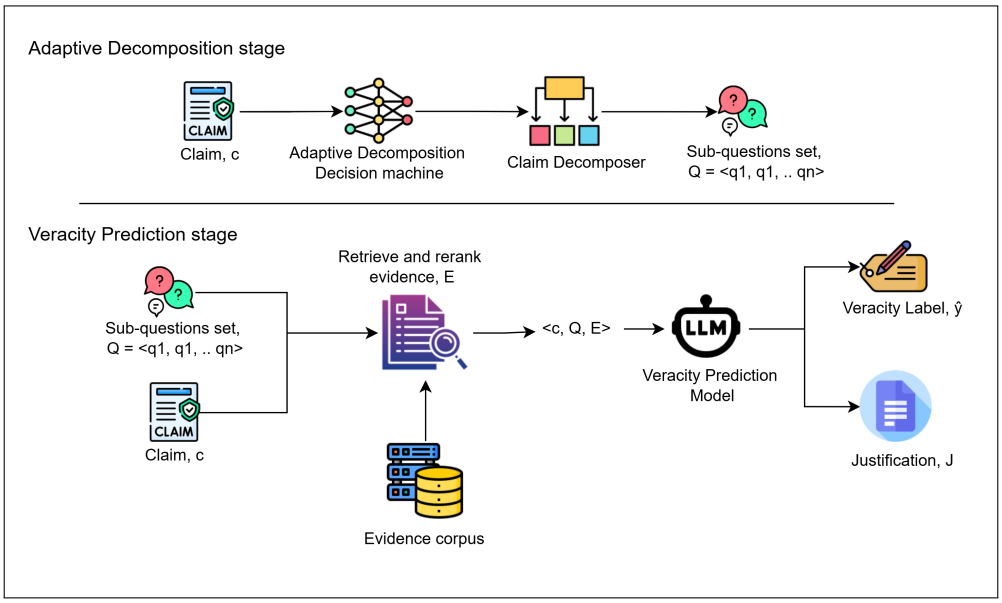


Figure 3.1: The fact-checking pipeline with adaptive decomposition.

3.2. Adaptive Decomposition

For adaptive decomposition experiments, an abstract concept called *complexity* is used to determine the extent of decomposition a claim requires. Two adaptive decomposition decision algorithms are proposed that predict the *complexity* of a claim. Both methods learn this concept from a supervised dataset. At test time, a claim is decomposed into the corresponding number of sub-questions depending on the complexity class the adaptive decomposer assigns to it.

3.2.1. Complexity dataset to train adaptive decomposition decision machines

To train the adaptive decomposition algorithms, the training dataset is designed by setting the minimum number of sub-questions a claim requires to correctly classify its veracity label as a proxy for its complexity. The language model used for veracity prediction of this annotation process is called the *feedback model*. The claims are assigned three increasing levels of complexity. Complexity level 0 corresponds to those claims which do not require any sub-questions to correctly classify them, level 1 corresponds to those claims which require one sub-question, and level 2 corresponds to those which require more than one. The claims that are still not classified correctly, even after generating all possible sub-questions are also assigned complexity level 2. The detailed algorithm is provided in Algorithm 1.

3.2.2. Adaptive Decomposition using latent representation of the claims

The first adaptive decomposition decision algorithm is inspired by the CTRLA paper [61], which hypothesizes that the latent representations of LLMs encode meaningful semantic properties. In order to extract the *complexity* property, a training corpus of claims $S_k = \{s_1, s_2, \ldots, s_n\}$ belonging to complexity level k is collected. These sentences are fed into an LLM to extract their token-level representations from each layer, which are hypothesized to capture semantic attributes like complexity [86, 87]. Let $\mathcal{F}_{\text{LLM}}$ denote a pretrained large language model with L layers and hidden size h. For an input sequence s_i consisting of t tokens, the token-level hidden representations across all layers are defined as:

$$\mathcal{F}_{\text{LLM}}(s_i) = \left\{ \mathbf{r}_{i,j}^{l} \in \mathbb{R}^h \,\middle|\, 1 \le j \le t,\ 1 \le l \le L \right\}, \tag{3.2}$$

where $\mathbf{r}_{i,j}^{l}$ denotes the hidden representation of the j^{th} token in sequence s_i at layer l.

Since the final token in a sequence attends to all preceding tokens [88], a pooling operation is applied whereby the hidden representation of the last token is selected to summarize the sequence at each layer. Let $\mathcal{P}$ denote this pooling function. For sequence s_i, the pooled representations across layers are defined as:

$$\mathbf{z}_i^l = \mathcal{P}(\{\mathbf{r}_{i,j}^{l}\}_{j=1}^{t}) = \mathbf{r}_{i,t}^{l} \quad \text{for } 1 \le l \le L, \tag{3.3}$$

Algorithm 1 Generate Complexity Training Data

Require: Claims set $[c_1 \dots c_m]$, Veracity labels $[y_1 \dots y_m]$, Verifier model M
Ensure: Complexity labels set $Complexity\_set$ with values $[k_1 \dots k_m]$

$Complexity\_set \leftarrow []$
for c_i in claim set $[c_1 \dots c_m]$ **do**
    Retrieve and rerank evidence E_i using c_i
    Predict label $\hat{y}_i = M(c_i, E_i)$
    if $\hat{y}_i = y_i$ **then**
        $k_i \leftarrow 0$
        $Complexity\_set \leftarrow Complexity\_set \cup \{k_i\}$
        continue
    end if
    solved $\leftarrow$ False
    while questions are generated **do**
        Retrive q_{ij} given c_i and previous questions $\langle q_{i1} \dots q_{ij-1} \rangle$
        Retrieve and rerank evidence E_i using c_i and sub-questions $\langle q_{i1} \dots q_{ij} \rangle$
        Predict label $\hat{y}_i = M(c_i, E_i, \langle q_{i1} \dots q_{ij} \rangle)$
        if $\hat{y}_i = y_i$ **then**
            $k_i \leftarrow \min(2, j)$
            solved $\leftarrow$ True
            break
        end if
        $j \leftarrow j + 1$
    end while
    if solved = False **then**
        $k_i \leftarrow 2$
    end if
    $Complexity\_set \leftarrow Complexity\_set \cup \{k_i\}$
end for

and the resulting set of pooled sequence-level representations is:

$$\mathcal{Z}_i = \left\{ \mathbf{z}_i^l \in \mathbb{R}^h \,\middle|\, 1 \le l \le L \right\}. \tag{3.4}$$

After processing all statements in complexity set S_k, Principal Component Analysis (PCA) is applied to the pooled representations $\left\{ \mathbf{z}_i^l \in \mathbb{R}^h \,\middle|\, 1 \le i \le n \right\}$ for each layer l. The first principal component for layer l of complexity level k is extracted as:

$$\mathbf{u}_k^l = \text{PCA}_1 \left(\left\{ \mathbf{z}_i^l \right\}_{i=1}^n \right), \tag{3.5}$$

resulting in a set of complexity representation vectors:

$$\mathcal{U}_k = \left\{ \mathbf{u}_k^l \right\}_{l=1}^L. \tag{3.6}$$

This process is repeated for each complexity class, resulting in training representations $\mathcal{U} = \{\mathcal{U}_k\}_{k=1}^{|K|} = \{\{\mathbf{u}_k^l\}_{l=1}^{L}\}_{k=1}^{|K|}$, where $|K|$ is the total number of complexity classes (three in this case).

For a given test claim c, its latent representation is extracted from the same LLM and pooled at each layer to obtain:

$$\mathcal{C} = \left\{ \mathbf{c}^l \in \mathbb{R}^h \,\middle|\, 1 \leq l \leq L \right\}. \tag{3.7}$$

To determine its corresponding complexity level, cosine similarity (Section A.3.1) is computed between the test claim's representation and each complexity's representations across all layers. This yields a similarity matrix:

$$\mathcal{S} = \left\{ \mathbf{s}^l \in \mathbb{R}^{|K|} \,\middle|\, \mathbf{s}_k^l = \cos\left(\mathbf{c}^l, \mathbf{u}_k^l\right),\ 1 \leq k \leq |K|,\ 1 \leq l \leq L \right\}, \tag{3.8}$$

where $\mathbf{s}_k^l$ denotes the cosine similarity between the test claim and complexity class k at layer l.

For each layer, the complexity class with the highest similarity is selected:

$$\hat{k}^l = \arg\max_k \ \mathbf{s}_k^l, \tag{3.9}$$

and majority voting across all $\hat{k}^l$ determines the final predicted complexity class. Figure 3.2 illustrates the flow of latent representation extraction and complexity assignment.

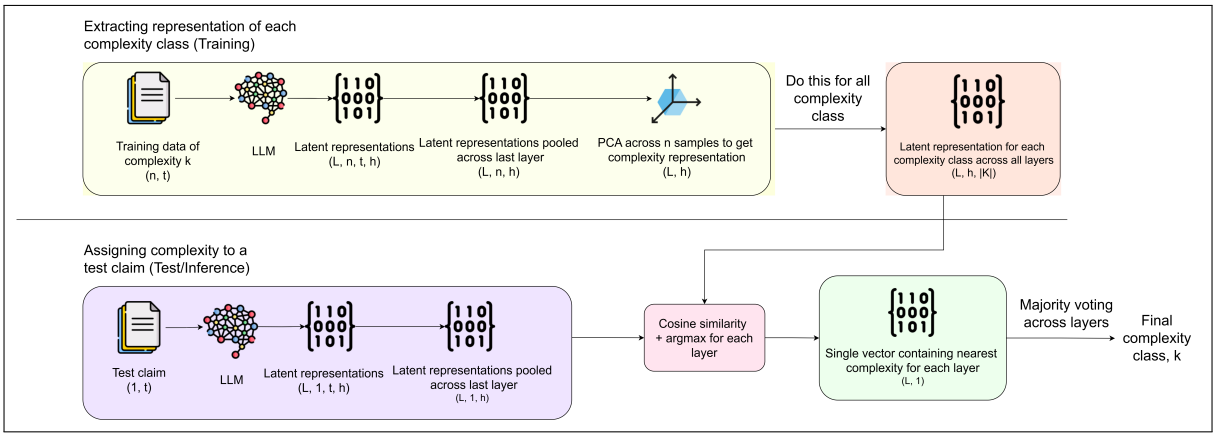


Figure 3.2: Training and inference of complexity assignment pipeline using latent representation-based method for adaptive decomposition.

3.2.3. Adaptive Decomposition using setfit

The second adaptive decomposition method involves finetuning a sentence transformer model [89] to learn the semantic difference between contrastive examples [90]. The sentence trans-

former backbone is paired with a classifier to assign a complexity class to the input claims. To fine-tune the model, positive and negative pairs of claims are created with similar complexity claims consisting of positive labels and different complexity claim pairs for negative labels. Cosine similarity Loss[1] is used to update the parameters of the sentence transformer backbone. After training the backbone, the classifier is trained to classify complexity classes which is a simple logistic regression algorithm. Figure 3.3 illustrates the training workflow.

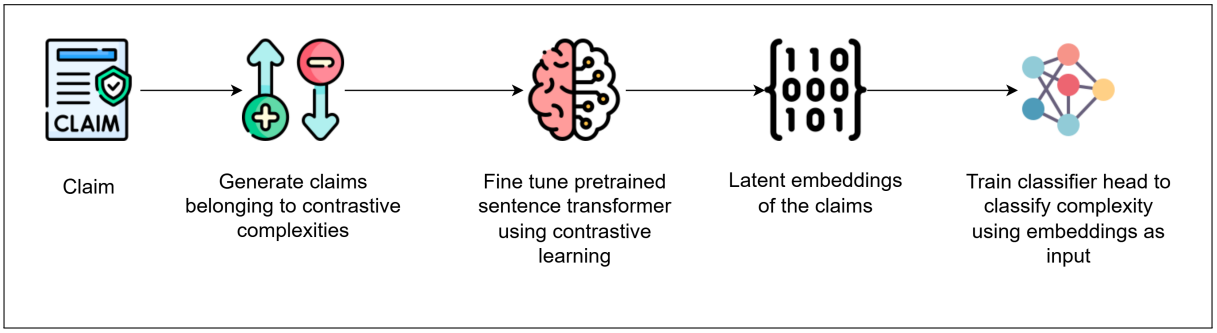


Figure 3.3: Training of setfit-based complexity assignment pipeline for adaptive decomposition.

3.3. Best of N sampling using a reward model

To train the reward model for selecting the best among N candidate output sequences, we draw inspiration from search against verifier methods. This approach is part of a test-time scaling strategy, where the reward model is trained to identify the most appropriate output from multiple independently generated sequences for the same input prompt. For this task, a claim (c) and its evidence (E) - with true veracity label y - is fed to a generator model[2] M to get its output sequence consisting of predicted verdict $\hat{y}$ and the corresponding justification J, governed by equation 3.1. The sub-question set is empty in these experiments, as they are run without decomposition. This setup allows the reward model to learn a preference for justifications that address all relevant aspects of the claim without relying on external sub-questions.

For the same (claim, evidence) pair, multiple independent output sequences are sampled, giving a set of independent (verdict, justification) sequences - $\{(\hat{y}^1, J^1), (\hat{y}^2, J^2), ..., (\hat{y}^N, J^N)\}$. Each of these sequences are given a score by a reward model. The reward model is a pre-trained language model with a classifier head that is fine-tuned to output a logit score $RM(c, \hat{y}^n, J^n)$ for the nth output sequence. The objective of the fine-tuning is to assign a higher score to a more appropriate output sequence. To do so, the reward model is fine-tuned with a binary cross-entropy loss function:

$$\mathcal{L}_{RM} = -[r^n \cdot \log RM(c, \hat{y}^n, J^n) + (1 - r^n) \cdot \log(1 - RM(c, \hat{y}^n, J^n))] \tag{3.10}$$

[1] https://pytorch.org/docs/stable/generated/torch.nn.CosineEmbeddingLoss.html

[2] The inference model used to generate veracity label and justification is referred to as a generator model in this experiment to align with the literature and avoid confusion with the reward model

where r^n is 1 if $\hat{y}^n$ = y and 0 otherwise. In other words, the reward model learns to give higher scores to the output sequence (y^n, J^n) pair if it leads to the correct veracity label y for that claim. The final verdict and justification ($\hat{y}^*$, J^*) is the output sequence that maximizes the reward model score:

$$n^* = argmax_{n \in \{1,...,N\}} RM(c, \hat{y}^n, J^n) \tag{3.11}$$

$$(\hat{y}^*, J^*) = (\hat{y}^{(n^*)}, J^{n^*}) \tag{3.12}$$

This approach of choosing the highest-scoring candidate from a set is referred to as *Best-of-N sampling* [26]. The workflow is illustrated with an example in Figure 3.4.

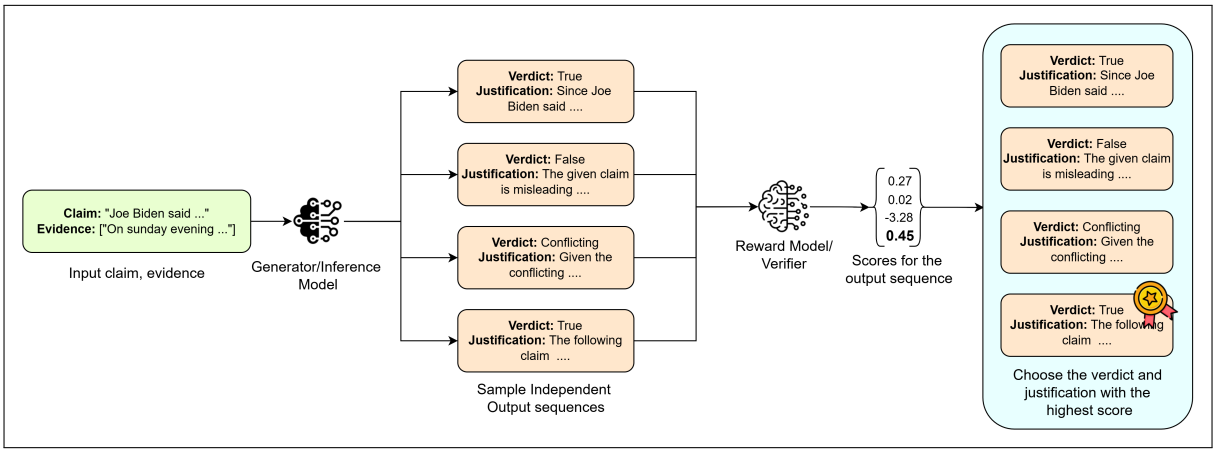


Figure 3.4: Workflow diagram highlighting search against verifier with an example.

4

Implementation Details and Experiments

4.1. Dataset

The experiments are run on the QUANTEMP [3] dataset which consists of numerical claims prepared by scraping real-world claims from fact-checking websites. These claims are inherently complex as they are considered claim-worthy by human fact-checkers. The original labels assigned by human fact checkers are mapped to three distinct labels - *True*, *False* and *Conflicting* through qualitative analysis of the context in which those labels were assigned. The dataset contains 9935 training samples, 3084 validation samples and 2495 test samples. Veracity class distribution of the samples is presented in Table 4.1

Dataset split	True	False	Conflicting
Training	1824	5770	2341
Validation	617	1795	672
Test	474	1423	598

Table 4.1: Distribution of Veracity Classes of the Quantemp dataset

4.2. Evidence retrieval and re-ranking

The evidence corpus is constructed following the retrieval pipeline outlined in the QUANTEMP framework [3]. Documents are collected from web search engine results using the claim as the primary query. For claim decomposition experiments, the generated questions are also used as queries alongside the claim. From this initial collection, the top 300 documents are retrieved using BM25 [91]. For claim decomposition run, the top 100 documents for each claim/sub-question are retrieved. To eliminate redundancy, these retrieved documents are deduplicated based on Levenshtein distance [92], implemented by `fuzzywuzzy` library [93, 94]. Pairs of documents exhibiting a similarity score above 80% are considered to be duplicates. Finally, this evidence is re-ranked using the claim to select the top 10 evidence for inference. The re-ranking is based on cosine similarity A.3.1 of their latent representations embedded by paraphrase-MiniLM-L6 sentence transformer [89]. For claim decomposition experiments, the

final selection of evidence ensures an equal allocation of top-*k* documents per query (i.e., claim and sub-questions), such that the total number of re-ranked evidence per instance remains consistent at 10.

4.3. Claim Decomposition

For each claim, sub-questions are generated similar to the methodology described in CLAIMDECOMP [13] by prompting the `gpt-3.5-turbo` model[1]. A key constraint imposed upon these sub-questions is that they must be answerable in *yes/no* format. This choice is supported by empirical findings which demonstrate the effectiveness of this format in improving accuracy [13, 3]. Moreover, this format enables the inference model to perform implicit verification of individual sub-questions and facilitate implicit aggregation of their answers to derive a final verdict. In addition to being answerable in a binary format, the generated sub-questions are required to satisfy several quality criteria - they must exhibit comprehensive coverage of the original claim, avoid redundancy, maintain clear grammatical and semantic coherence, and remain directly relevant to the claim. The exact system prompt used to elicit these sub-questions from the language model is presented in Figure 4.1.

You are tasked with generating a set of questions to break down and evaluate the veracity of a given fact-checking claim. Follow these strict guidelines when creating the subquestions:

1. **Comprehensive Coverage:** The subquestions should comprehensively address all relevant aspects of the claim to enable a thorough fact-check.
2. **Non-redundancy:** Ensure there is no overlap between the subquestions. Each question should be unique in meaning and not semantically repetitive.
3. **Relevance:** Every question must be relevant in determining the veracity of the claim. Avoid irrelevant or tangential questions.
4. **Clarity and Grammar:** Write questions that are clean, concise, and grammatically correct to maintain clarity.
5. **Yes/No Format:** Each question must be strictly answerable with a simple ”Yes” or ”No” response.

Figure 4.1: System prompt to generate sub-questions given a claim, adhering to the standards defined in the CLAIMDECOMP paper.

4.4. Hardware configuration

The experiments were conducted on a dedicated workstation running Arch Linux `x86_64` with kernel version 6.13.7-arch1-1. The system featured an AMD EPYC 7302P processor with 32 physical cores operating at a base frequency of 3.0 GHz. For GPU-accelerated tasks, the workstation was equipped with two NVIDIA GeForce RTX 3090 graphics cards, each offering substantial parallel processing capabilities optimized for compute-intensive workloads. The machine had 256 GiB of system memory. A total of 654 packages were managed via the

[1] https://platform.openai.com/docs/models/gpt-3.5-turbo

`pacman` package manager, and the shell environment was configured using `zsh` version 5.9. The graphical output was configured at a resolution of 1024×768, although experiments were primarily conducted in terminal-based sessions.

The `llama-3.2-3B` for decomposition experiments was fine-tuned on google colab[2] which is hosted on Google Compute Engine, running Ubuntu 22.04.4 LTS `x86_64` with kernel version 6.1.123+. The system was provisioned with an Intel Xeon processor featuring 12 virtual cores operating at a base frequency of 2.2 GHz. The instance utilized an NVIDIA A100 SXM4 with 40 GiB of dedicated GPU memory. The machine was equipped with approximately 85 GiB of system memory. All software dependencies were managed within a containerized environment and code execution was carried out through a Jupyter notebook interface [95].

4.5. Implementation Details

This section presents implementation details of the methods studied in the thesis. Sub-section 4.5.1 outlines the two adaptive decomposition approaches, including the models and tools used, training procedure (if any), and the baseline methods used for comparison. Sub-section 4.5.2 describes the construction of reward model training data, the reward model's architecture and training details, and the baseline methods against which it is evaluated.

4.5.1. Adaptive Decomposition

Llama-3.1-8b-Instruct [96] is employed as the inference model across both the claim decomposition and adaptive decomposition experiments. In addition to these methods, a baseline experiment *without decomposition* is conducted to enable direct comparison of these methods. All inference is performed in a few-shot setting, where the two most semantically similar example claims are retrieved from the validation split of the QUANTEMP dataset to serve as prompts. Similarity is computed using cosine similarity over the latent embeddings of the claims. The inference model is configured with a temperature of 0.3 and nucleus sampling using a top-p value of 0.95. The maximum number of output tokens is limited to 512. The framework for these experiments is implemented in Python with Huggingface's transformer library [97]. The system prompt used for both *without decomposition* and *with decomposition* inference settings, and the structure of the user prompt are illustrated in figure A.1, figure A.2 and figure A.3 respectively.

To train the complexity classification for the adaptive decomposition framework, a total of 1500 samples were annotated. These samples were stratified from the training set to ensure balanced distribution over the veracity labels. Complexity class annotations were generated using two separate large language models as feedback sources: (i) `LLaMA-3.1-8B-Instruct` and (ii) `GPT-3.5-Turbo`. In the results section, we refer to these two annotation strategies as *llama feedback* and *gpt feedback*, respectively. The distribution of complexity classes produced by each feedback model is summarized in Table 4.2.

[2] https://colab.research.google.com/

Feedback Method	Class 0	Class 1	Class 2
llama Feedback	807	135	558
gpt Feedback	878	102	520

Table 4.2: Distribution of samples across complexity classes for different feedback methods.

For **latent representation-based** adaptive decomposition approach, we utilize *Mistral-7B-Instruct* [98] to extract the latent embeddings for both training and the test claims. Principal Component Analysis algorithm is implemented using scikit-learn [99] library with n_components set to 1. The resulting principal component is interpreted as the primary complexity direction, which serves to differentiate between complexity classes. In the **SetFit-based** adaptive decomposition method, we employ the `all-mpnet-base-v2` model [89] as the embedding backbone, combined with a logistic regression classifier to predict complexity classes. The classifier is trained for 500 steps with a batch size of 45. An *oversampling* strategy is used during training to balance positive and negative samples at each step, thereby addressing class imbalance within the training data.

In addition to `llama-3.1-8B-Instruct`, adaptive decomposition experiments are conducted using three additional language models, each representing a distinct model category. The first is `gpt-3.5-turbo` [100], a proprietary model with significantly stronger performance. The second is `deepseek-r1-7B` [66], a model specifically designed for step-by-step reasoning and multi-hop inference tasks. The third is a fine-tuned version of `llama-3.2-3B` [96], trained under both *with decomposition* and *without decomposition* configurations using the training split of the QUANTEMP dataset. For both training and inference in these experiments, the top five re-ranked evidence documents are used as input to the model.

The model is fine-tuned using the Parameter-Efficient Fine-Tuning (PEFT) framework [101], specifically by integrating Low-Rank Adaptation (LoRA) adapters into each attention block of the base model. The LoRA adapters are configured with a rank of 64 and a scaling factor (α) of 16. Fine-tuning is performed for 1,241 steps with a batch size of 8. Optimization is carried out using the `paged_adamw_32bit` optimizer, with a learning rate of 2×10^{-4}, following a cosine learning rate decay schedule with a warmup ratio of 3%. To manage GPU memory constraints, gradient checkpointing is enabled throughout training. Additionally, gradient clipping is applied with a maximum norm of 0.3 to ensure stable optimization. Mixed-precision training (FP16) is employed to improve memory efficiency and training speed. No dropout is applied to the LoRA adapter layers during fine-tuning.

4.5.2. Test time scaling using Reward model

Due to resource constraints, we employ mixed-precision inference using `llama-3.1-8B-Instruct` to generate 10 output sequences per claim. This variant of the model is provided by `ollama`[3] library. The same model is also used for the majority voting baseline and the top-1 decod-

[3] `https://ollama.com/`

ing baseline. These output sequences are generated with a temperature value of 0.45 and nucleus sampling with the top-p value of 0.95. To train the reward model, a total of 3,500 claims from the QUANTEMP training set are sampled. For each claim and retrieved evidence pair, 10 output sequences are generated. Duplicate reasoning paths are removed to prevent overfitting. A binary label is assigned to each unique reasoning path: a label of 1 is given to paths that yield a correct veracity prediction, while a label of 0 is assigned to incorrect or hallucinated reasoning paths. In some cases, sequences containing plausible but hallucinated reasoning are explicitly retained and labeled as 0 to help the model learn fine-grained distinctions in reasoning quality. This annotation process resulted in 12,975 training samples and 3,244 validation samples. The distribution of target labels in both the training and validation sets is presented in Table 4.3.

Distribution	Class 0	Class 1
Training	8885	4090
Validation	2221	1023

Table 4.3: Distribution of samples across reward labels for training and validation set of reward models.

The reward model architecture is based on the `llama-3.2-3B` model, extended with a classification head. Leveraging a decoder-based architecture enables the model to better capture linguistic nuances and contextual dependencies. Fine-tuning is performed using the Parameter-Efficient Fine-Tuning (PEFT) framework [101], with LoRA adapters configured using a rank of 8 and a scaling factor (α) of 16. The model is fine-tuned for 3 epochs with a batch size of 32, using the AdamW optimizer with $\epsilon = 1 \times 10^{-8}$ and an initial learning rate of 1×10^{-3}. A linear learning rate scheduler without warm-up steps is applied, gradually decreasing the learning rate throughout the training process.

4.6. Evaluation metrics

4.6.1. Quantitative metrics

The performance of claim veracity prediction is evaluated using the F1-score (section A.3.2). In addition to the overall F1-score, per-class F1-scores are computed for each veracity label to assess the impact of different methods and models across specific classes. The F1-score is chosen as the primary evaluation metric because it accounts for both false positives and false negatives, and is particularly suitable in the presence of class imbalance, as observed in the QUANTEMP dataset (Table 4.1). To provide a comprehensive view of model performance, both macro and weighted F1-scores are reported. While the macro F1-score treats all classes equally, the weighted F1-score incorporates class frequencies, offering a more balanced perspective on the overall predictive capability of the models.

4.6.2. Grounded theory-based qualitative analysis to identify error taxonomies

To better understand the outcomes of our experiments, we conduct a qualitative analysis of the outputs generated by the language models. This analysis is informed by principles from grounded theory [102], which offers a structured approach to qualitative data interpretation. In particular, we employ the *constant comparison method* [103], which involves iterative comparison of individual instances to identify patterns and organize them into coherent categories.

In our study, each instance refers to an output sequence from the language models, comprising a predicted claim verdict $\hat{y}$ and its corresponding justification J. The resulting categories serve to reveal overarching patterns that explain the behavior of the models. These categories may capture common error types, qualitative improvements achieved by specific methods, or emergent reasoning strategies.

While inspired by established methodologies, our approach is adapted to the specifics of the claim verification task and does not strictly adhere to the original procedures outlined in the literature. A high-level overview of our taxonomy generation process is illustrated in Figure 4.2, and the detailed algorithm is presented in Section A.4.

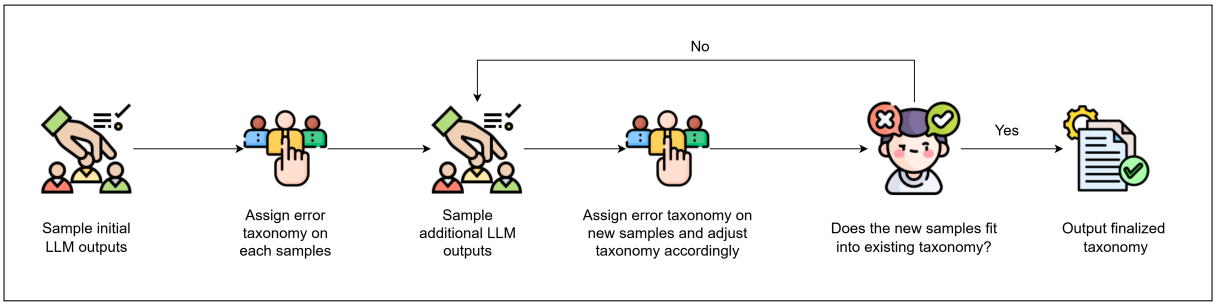


Figure 4.2: Flowchart of our qualitative analysis process to understand the output sequences produced by our inference models.

5 Results

5.1. Impact of Claim decomposition on veracity prediction

The results of the *without decomposition* and *with decomposition* configurations for veracity prediction performance are presented in Table 5.1. `llama-3.1-8b-Instruct` [96] is used as the inference model for this experiment. While the *False* class performance remains relatively stable, claim decomposition significantly affects the *True* and *Conflicting* classes. Specifically, decomposition improves *Conflicting* class performance by 26.8% relative to the without-decomposition baseline, while *True* class performance drops by 23%.

Method	Per Class F1			Overall F1	
	True	False	Conflicting	Macro	Weighted
Vanilla inferencing					
without Decomposition	**41.2**	66.7	30.6	46.2	53.2
with Decomposition	31.7 (–23.0%)	64.8 (–2.9%)	38.8 (+26.8%)	45.1 (–2.4%)	52.3 (–1.7%)
Adaptive Decomposition - Latent representation based classifier					
Llama feedback	35.9 (–12.9%)	**67.8** (+1.7%)	**39.4** (+28.8%)	**47.7** (+3.3%)	**54.6** (+2.6%)
gpt feedback	41.5	67.6	32.0	47.0	54.1
Adaptive Decomposition - setfit classifier					
Llama feedback	38.0	66.4	35.4	46.7	53.6
gpt feedback	34.9	67.8	38.3	47.0	54.5
Adaptive Decomposition - upper bound					
Decompose Conflicting	43.6	69.0	46.2	52.9	58.7

Table 5.1: Performance of llama inference model with various decomposition settings. The *llama feedback* and *gpt feedback* refer to the corresponding adaptive decomposition methods trained on a complexity dataset annotated by respective models. The relative gains are computed w.r.t the *without decomposition* baseline.

Qualitative analysis on the results indicates that the presence of sub-questions helps guide in correctly reasoning the various ambiguous aspects of Conflicting claims. Figure 5.1 demonstrates this with an example. With claim-decomposition, the sub-question detailing the aspect of **"double offshoring jobs"** allows the inference model to be more precise in its reasoning against the evidence along that aspect. This helps the model to acknowledge that the

evidence shows a high number of offshore contracts but doesn't directly support this **double** aspect, thus correctly predicts the veracity as *Conflicting*. When inferred without decomposition, the model misses this part, as it is embedded in a larger claim and only reasons along the general sentiment of the claim, misclassifying the claim as *True*.

[Claim]: Says Donald Trump "let federal contractors double offshoring jobs in his first 18 months in office."

[Subquestions]:

1. Did the number of offshoring jobs increase during Donald Trump's first 18 months in office?
2. Did Donald Trump allow federal contractors to double offshoring jobs?

[Label]: **Conflicting**

Without Claim Decomposition Inference

[Prediction]: **True**

[Reasoning]:
The evidence provided supports the claim that Donald Trump ... The evidence from Politifact ... engaged in offshoring, with 28% of the top 50 federal contractors and 25% of the top 100 having offshored American jobs during his term. Additionally, ... one-in-four taxpayer dollars spent on federal procurement contracts went to firms offshoring jobs during his presidency, which is at least $425.6 billion. This suggests ... even increase.

With Claim Decomposition Inference

[Prediction]: **Conflicting**

[Reasoning]:
The evidence provided does not directly support the claim that Donald Trump ... While it is true that a significant number of federal contractors engaged in offshoring during Trump's presidency, with 28% of the top 50 federal contractors and 25% of the top 100 federal contractors being certified ... the evidence does not provide a clear indication of whether the number of offshoring jobs doubled during Trump's first 18 months in office.
Moreover, ... Trump "let federal contractors double offshoring jobs" is an assertion that is not supported by the provided evidence. The evidence ... 18 months of his term.

Figure 5.1: Reasoning of a *Conflicting* claim without claim decomposition and with claim decomposition, highlighting how decomposition enable the model to reason along various aspects of the claim precisely.

On the other hand, the presence of sub-questions due to decomposition causes the model to overthink which leads several *True* class claims to be mislabeled to either *False* or *Conflicting*. We term this phenomenon as ***decomposition induced overthinking*** which is a part of a larger taxonomy of artifacts commonly found in language models [18, 19]. Another related taxonomy is ***contextual overthinking*** which is discussed in section 5.5. To demonstrate *decomposition induced overthinking*, we present an example in Figure 5.2. The model's reasoning, under the decomposition setting, reasons about aspects of whether the budget cut proposed by Sen. Rand Paul was implemented. This can be attributed to the second sub-question, which

elaborates on the aspect of **"whether there was indeed a reduction"**. The claim did not require verifying this aspect as it only needed to fact-check whether there was a cut included in the proposal made by Sen. Rand Paul. The *without decomposition* setting avoids such overthinking and only reasons along the relevant aspects of the claim.

[Claim]: Says Sen. Rand Paul's 2011 budget "included a big cut in the CDC."

[Subquestions]:

1. Did sen. Rand Paul's budget proposal in 2011 include a cut in the cdc's funding?
2. Was there a significant reduction in the cdc's funding in sen. Rand Paul's proposed budget in 2011?
3. Did sen. rand paul propose a budget in 2011?

[Label]: **True**

Without Claim Decomposition Inference

[Prediction]: **True**

[Reasoning]:
The evidence provided shows that Sen. Rand Paul's 2011 budget did include a significant cut in the CDC ... FY2012 and FY2013 budgets called for a 20 percent cut for the CDC ... While the evidence does not provide a specific dollar amount for the CDC cut in the 2011 budget, it is clear that Sen. Paul's budget did include a cut to the CDC.

With Claim Decomposition Inference

[Prediction]: **Conflicting**

[Reasoning]:
1. The evidence confirms that Sen. Rand Paul did propose a budget in 2011. The evidence states ... in 2011."
2. The evidence also confirms that Sen. Rand Paul's budget proposal in 2011 included a significant reduction in the CDC's funding. The ... for a 20 percent cut for the centers for disease control and prevention."
3. The evidence does not provide a clear answer to whether the CDC's funding was actually reduced as a result of Sen. Rand Paul's budget proposal in 2011. While the proposal called for a 20 percent cut, it is unclear whether this cut was implemented.
Therefore, the claim that Sen. Rand Paul's 2011 budget "included a big cut in the CDC" is conflicting, as ... CDC's funding is unclear.

Figure 5.2: Reasoning of a *True* claim without claim decomposition and with claim decomposition, highlighting the model's *decomposition induced overthinking*.

Analyzing the error taxonomies of misclassifications in True class (under decomposition setting) indicates that *decomposition-induced overthinking* is the leading cause of the performance hit (Table 5.2) taken when inferred with decomposition. Another key error is the presence of noise in the re-ranked evidence, which can be solved by a better retrieval algorithm. Detailed descriptions of the two errors mentioned in Table 5.2 - *decomposition induced overthinking* and *noisy re-ranked evidence*, are elaborated in Table A.1.

5.2. Adaptive decomposition helps mitigate issues introduced by decomposition while preserving its benefits

Table 5.1 also shows the result of the two adaptive decomposition methods trained on the two variations of feedback models. Results indicate that the adaptive decomposition methods preserve the advantage of guided reasoning (for Conflicting claims) while balancing the issues introduced for True claims that came with decomposition. This is achieved by selectively decomposing certain claims considered worthy of decomposition. Importantly, this selective strategy does not disrupt the performance of the *False* class, which remains relatively stable across all settings. The performance is reflected in the respective True and Conflicting F1 scores values for adaptive decomposition settings, which lie between the values obtained when inferred with and without decomposition settings. This shows that decomposing based on the complexity of a claim is an effective strategy to correctly decide the claims that require decomposition. Among these methods, the latent representation-based classifier trained on the samples annotated by `llama-3.1-Instruct-8B` model (called llama feedback) outperforms the rest, achieving the macro and weighted F1 scores of 47.7 and 54.6 respectively.

However, it fails to achieve the theoretical upper bound of macro F1 score of 52.9 and weighted F1 score of 58.7, which is obtained by decomposing only claims with *Conflicting* labels. This highlights the inability of the notion of *Complexity* to directly capture nuances present in the theoretical upper-bound. On the other hand, it is not ideal for our notion of complexity to directly align with the upper-bound as it is directly based on the veracity labels. In other words, using the veracity label signal directly to design a decision machine for decomposition is not ideal as it will overfit to the problem (we might as well directly fine-tune a veracity prediction model). Better nuances might be captured with an alternative definition of *Complexity* (apart from the one used in algorithm 1), a larger training dataset, or a stronger decision model.

Error Taxonomy	**Frequency**
Decomposition induced overthinking	24 (60%)
Noisy re-ranked Evidence	12 (30%)
Others	4 (10%)

Table 5.2: Taxonomy of errors found in the *True* class while inference with Decomposition. This result is obtained by qualitatively analyzing 40 random samples, providing insights into common mistakes made by the model in a decomposition setting.

5.3. Impact of decomposition and adaptive decomposition on different models

To investigate whether the impact decomposition has on the per-class performances translates to diverse language models, additional adaptive decomposition experiments are run on a stronger proprietary model - `gpt-3.5-turbo`[1], a thinking model - `deepseek r1` [66], and a

[1] https://platform.openai.com/docs/models/gpt-3.5-turbo

model fine-tuned for the task - `llama-3.2-3B`[2]. Table 5.3 highlights the effect decomposition and adaptive decomposition has on these models. The best-performing adaptive decomposition method from Table 5.1 (latent representation based on llama feedback) is used for these experiments.

Method	Per Class F1			Overall F1	
	True	False	Conflicting	Macro	Weighted
llama-3.1-8b					
without Decomposition	**41.2**	66.7	30.6	46.2	53.2
with Decomposition	31.7	64.8	38.8	45.1	52.3
Adaptive decomposition	35.9	**67.8**	**39.4**	**47.7**	**54.6**
gpt-3.5-turbo					
without Decomposition	41.3	**72.6**	3.2	39.1	50.0
with Decomposition	**41.9** (+1.5%)	71.9 (–1.0%)	**25.4** (+693.8%)	46.4	55.1
Adaptive decomposition	41.4 (+0.2%)	71.7 (–1.2%)	19.6 (+512.5%)	44.2	53.5
deepseek r1 7b					
without Decomposition	39.4	58.8	28.8	42.3	47.9
with Decomposition	**40.4** (+2.5%)	**60.2** (+2.4%)	29.3 (+1.7%)	43.3	49.0
Adaptive decomposition	38.7 (–1.8%)	59.3 (+0.8%)	**31.0** (+7.6%)	43.0	48.6
llama 3b fine tuned					
without Decomposition	43.3	79.2	31.4	51.3	60.9
with Decomposition	37.0 (–14.5%)	**79.7** (+0.6%)	40.0 (+27.4%)	52.3	62.1
Adaptive Decomposition	41.7 (–3.7%)	**79.6** (+0.5%)	39.9 (+27.1%)	53.7	62.9

Table 5.3: Comparison of different state-of-the-art models in different decomposition settings. Adaptive decomposition corresponds to using representation learning, which is the best-performing method in Table 5.1. The relative gains and hits are computed w.r.t the *without decomposition* baselines of the respective methods.

For `gpt-3.5-turbo`, performance is stable for *True* and *False* classes, but decomposition helps for *Conflicting* class. Due to this stable performance, adaptive decomposition is not of much help over full decomposition. This is in alignment with the findings in the *decomposition dilemma* paper [17], which states that decomposition is not helpful for stronger models. On the other hand, *Conflicting* labels are artificially mapped while creating the Quantemp dataset [3], and they encompass various cases such as insufficient evidence, misleading evidence, exaggerated claims, etc. Since it is not explicitly defined anywhere on the internet, it is unlikely for the model to learn this notion from its pre-training or post-training data. Hence, the presence of sub-questions guides the reasoning in these cases thus leading to its improved per-class F1 score.

For `deepseek-r1`, decomposition leads to a slight improvement in performance across all classes. However, the observed gains are not substantial enough to draw any strong conclusions. Adaptive decomposition shows a modest improvement for *Conflicting* class, with a

[2]https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/

relative gain of 7.6% compared to the *without decomposition* configuration, and 5.9%[3] relative to the *with decomposition* configuration. It is noteworthy that the task of fact-checking remains a challenging problem, even for state-of-the-art reasoning models such as `deepseek-r1`, which have demonstrated competitive performance across a variety of tasks [66].

The motivation for fine-tuning the model is to investigate whether the performance trends observed with decomposition and adaptive decomposition generalize to a fine-tuned model. The results follow a similar pattern to those observed with the `llama-3.1-8B-Instruct` model, with decomposition yielding improvements in the performance of the *Conflicting* class, while slightly reducing performance in the *True* class. Adaptive decomposition, however, effectively balances this trade-off by strategically decomposing certain claims, leading to improved overall performance.

Key Insight (RQ1 and RQ2)

1. *Claim decomposition improves performance on the **Conflicting** class by guiding the model's reasoning, while the **True** class takes a hit due to decomposition-induced overthinking and Noisy re-ranked evidence.*
2. *Adaptive decomposition helps balance the issue of overthinking in smaller general-purpose models and fine-tuned models. However, this gain is not translated into a stronger proprietary model and thinking model.*

5.4. Selecting the best solution using a reward model improves veracity prediction performance.

The performance of majority voting and Best of N samples using a reward model along with top 1 decoding as the baseline method is presented in Table 5.4. The results are obtained by sampling 10 independent output sequences for a given input from the generator model. For these experiments, the generator model is fixed to `llama-3.1-8b-Instruct` as it gives competitive and reliable results in the previous experiments. However, a mixed precision variant is employed for these experiments owing to resource constraints. Though not explicitly discussed in our results due to resource constraints, we hypothesize that other generator models will also give similar performance gains (over the different methods).

Out of all the methods, the best of N sampling runs achieves the best performance across all the veracity classes. This shows that the reward model is capable of choosing the reasoning sequence that leads to the correct verdict, despite the generator model not favoring those sequences from its solution space. This is in alignment with other studies that observed an improved performance when performing solution path selection with a reward model [20, 23]. The findings from a detailed qualitative investigation (section 5.5) further solidified our intuition that the reward model learns to favor output sequences that has better reasoning during its training. The reward model helps navigate the possible solution paths and ranks them aiding

[3] This computation is not shown in the table.

Method	Per Class F1 Score			Overall F1 Score	
	True	False	Conflicting	Macro	Weighted
Top 1 Decoding	33.5	66.4	34.7	44.8	52.5
Majority Voting	34.9	68.6	35.1	46.2	54.2
Best of N sampling with RM	**44.2** (+31.9%)	**75.3** (+13.4%)	**40.3** (+16.1%)	**53.2** (+18.8%)	**61.0** (+16.2%)
Theoretical upper bound	63.9	85.1	69.6	72.9	77.4

Table 5.4: Reasoning performance of the models. Here, the number of samples for majority voting and best of N sampling is set to 10. The generator model is llama-3.1-Instruct-8b, and the reward model is fine-tuned llama-3.2-3b. The gains method is computed w.r.t the *top 1 Decoding* baseline.

in selecting the best path suited for the claim.

However, the reward model's performance is unable to reach the theoretical upper-bound which is created by assigning a claim to be correctly predicted if at least one of its output sequences leads to the correct label. We hypothesize that this gap could be further narrowed by employing a stronger reward model and incorporating additional annotated training samples.

Another interesting observation is that the majority voting inference achieves better performance over sampling the generator LLM only once (top 1 decoding). This observation is in alignment with the literature that concluded that sampling only one output sequence often produces unreliable results [20, 21]. This indicates that we can improve the performance of fact-checking using LLM with a tradeoff of increasing test time compute.

5.5. Reward model chooses output sequence with better reasoning and lenient evaluations

To further investigate how the Best of N sampling combined with the reward model leads to improved performance, we conducted a qualitative analysis on samples where Best of N sampling succeeds while majority voting fails. Since majority voting involves multiple output sequences, we selected a reasoning trace from an output sequence that results in the majority verdict for comparison. Following the methodology outlined in Section 4.6.2, we analyzed 40 samples to define a taxonomy of improvements in the output sequences selected by the reward model.

One of the major improvements is in the quality of the reasoning in the output sequence chosen by the reward models. It covers 52.5 % of the improvement observed (Table 5.5). Though it is a broad category, it covers a series of related patterns such as a better understanding of numerical figures and their implications, being robust to noisy evidence in giving a verdict, and assigning a verdict that aligns with the reasoning. One such example is given in Figure 5.3, in which the reward model chooses an output sequence that has better comprehension of the numerical figures present in the evidence. In the example, the claim states that the remittance figures are **over 400 billion Kenyan shillings in 2021**. In the evidence, there are two figures - one mentioning the exact figure while another mentions a figure of **USD 3,718 million.** With majority voting, the verdict(s) are assigned to *Conflicting* due to these two

[Claim]: Diaspora remittances "now stand at an all-time high of over KSh400 billion in 2021".

[Evidence]:

1. as a consequence, diaspora remittances have increased tenfold over the last decade; and they now stand at an all-time-high at over ksh. 400 billion per annum as of 2021, surpassing some of our traditional exports as foreign exchange earner.
2. diaspora remittances were robust in 2021, with total remittances in 2021 reaching a record usd3,718 million. of daily users

[Label]: **True**

Reasoning without Reward Model

[Prediction]: **Conflicting**

[Reasoning]:
The evidence provides conflicting information about the exact amount of diaspora remittances in 2021, and some sources mention figures that are not directly comparable to KSh400 billion.

Reasoning with Reward Model

[Prediction]: **True**

[Reasoning]:
The evidence supports the claim that diaspora remittances reached an all-time high of over KSh400 billion in 2021. However, there are minor variations in the wording and units used across different sources.

Figure 5.3: An example highlighting improved reasoning with output sequence chosen by the Reward model. It picks up the sequence that comprehend and correctly reason against nuanced numerical figures leading to the correct verdict.

differing units. However, if the exchange rate of Kenyan shillings to US Dollars in 2021[4] is to be considered, the two values are almost similar with USD 3,718 million roughly converting to about 408 billion Ksh. Since the claim states that the values are **over 400 billion Ksh**, the evidence supports the claim despite the differing units of representing the amount. This nuance in the numerical figure is picked up in the reasoning chosen by the reward model, and even acknowledges the minor variations in the units across the different sources.

Taxonomy of Improvements	Frequency
Improved Reasoning	21 (52.5%)
Mitigation of contextual overthinking	10 (25%)
Others	9 (22.5%)

Table 5.5: Taxonomy of improvements observed while decoding using selecting the best output sequence using a reward model over majority voting.

Another noteworthy category in the taxonomy of improvements is the mitigation of *contextual overthinking* in the reasoning selected by the model, which was observed in 25% of the an-

[4]http://exchangerates.org.uk/KES-USD-spot-exchange-rates-history-2021.html

[Claim]: Says the 1956 Republican Party platform supported equal pay, the minimum wage, asylum for refugees, protections for unions and more.

[Evidence]:

1. in 1928 the gop platform read: the party favors freedom in wage we support an equitable minimum wage for american workers one providing fair wages
2. in case youre not up on what the gop stood for in eisenhowers day, the 1956 republican platform pledged to provide federal assistance to low-income communities; protect social security; provide asylum for refugees; extend the minimum wage; improve the unemployment benefit system so it covers more people; strengthen labor laws so workers can more easily join a union; assure equal pay for equal work regardless of sex.

[Label]: **True**

Reasoning without Reward Model

[Prediction]: **Conflicting**

[Reasoning]:
The evidence provided supports many of the claims made in the original statement, but some quotes are from earlier years or mention specific policies, and not all points are explicitly stated in the 1956 platform.

Reasoning with Reward Model

[Prediction]: **True**

[Reasoning]:
The claim is supported by multiple quotes from the 1956 Republican Party platform, which mention equal pay, the minimum wage, asylum for refugees, protections for unions, and other issues. While there are some minor discrepancies and nuances, the overall evidence suggests that the claim is accurate.

Figure 5.4: An example highlighting leniency in contextual overthinking with the output sequence chosen by the Reward model.

alyzed cases (see Table 5.5). In these instances, the reasoning is evaluated with a degree of contextual leniency, allowing the model to arrive at the correct verdict despite subtle mismatches in the evidence. An illustrative example is shown in Figure 5.4, where the claim asserts that the **1956 Republican party supported several issues.** The retrieved evidence does not explicitly state this support; instead, it references support for **certain policies that correspond to those issues.**

This discrepancy leads many output sequences to classify the claim as *Conflicting*. The majority-voted reasoning also highlights a mismatch in dates, noting that similar support existed in 1928 for one of the issues, which ideally should be disregarded. In this context, fact-checking may benefit from a more lenient interpretation, acknowledging that the Republican party did support all the mentioned issues in 1956, albeit through various policy endorsements. The output selected by the reward model captures this nuance: it acknowledges the lack of explicit support but opts for a contextually lenient interpretation that ultimately leads to the

correct classification.

Unlike *decomposition-induced overthinking*, which requires mitigation (as discussed in Section 5.1), contextual leniency is a subjective concept. Its appropriateness depends heavily on the evaluative perspective. For example, an opposing political party might reasonably interpret such a claim as *False*, while a sympathetic organization might offer a more charitable interpretation. Our support for contextual leniency in this case is grounded in its ability to yield the correct veracity label for the task. Nevertheless, it is important to recognize that even these "correct" labels may be subject to biases introduced by the human fact-checkers who constructed the dataset [104, 105, 106].

Given the nuanced and potentially contentious nature of contextual leniency, we encourage future work to more thoroughly investigate its role, appropriateness, and broader implications in automated fact-checking. A detailed overview of the identified improvement categories is provided in Table A.2.

Key Insight (RQ3)

*Using a Reward Model to select the best output sequence improves veracity prediction performance by selecting sequences with better **(numerical) reasoning** and exhibiting **leniency over contextual-overthinking**.*

6

Conclusion

6.1. Conclusion

In this study, two widely adopted strategies for enhancing the reasoning capabilities of large language models (LLMs) were investigated within the context of fact-checking complex claims. The first approach examined is *claim decomposition*, which involves breaking down complex claims into simpler sub-claims or sub-questions to guide the reasoning process of the LLM. Experimental results indicate that this method is particularly effective for claims categorized as *Conflicting*, where the decomposition aids in clarifying ambiguous or contradictory elements. However, the use of decomposition introduces two significant challenges, especially in the case of claims labeled as *True*. The first issue, termed *decomposition-induced overthinking*, arises when the generated sub-questions lead the model to apply excessively strict evaluative criteria, ultimately resulting in incorrect veracity predictions. The second challenge pertains to the re-ranking of outputs based on sub-questions, which can introduce extraneous or noisy evidence. This, in turn, may divert the model's focus from the most relevant aspects of the original claim, thereby affecting overall fact-checking performance.

To address the limitations associated with claim decomposition, the concept of *adaptive decomposition* is introduced, wherein claims are selectively decomposed based on their estimated *complexity*. In this context, complexity is treated as an abstract attribute of each claim and is realized using a proxy measure: the minimum number of sub-questions required to correctly predict the claim's veracity label. This proxy enables a practical approximation of complexity in experimental settings.

Empirical findings indicate that *adaptive decomposition* effectively mitigates the adverse effects associated with full decomposition in *True* claims, while preserving the performance gains observed in *Conflicting* claims. These improvements are consistently observed in smaller open-source and fine-tuned models. However, the benefits do not generalize to larger proprietary or thinking models. Nonetheless, decomposition continues to offer advantages for the *Conflicting* class even in stronger proprietary models, suggesting that it remains a viable strategy for enhancing reasoning in language models.

Finally, the concept of test-time scaling was investigated to enhance reasoning capabilities

by employing a reward model trained to select the most appropriate output sequence from a set of candidate outputs generated by the fact-checking model. The application of the reward model not only improves veracity prediction performance but also enhances the overall quality of the reasoning sequences. Notably, the reward model helps select better reasoning traces and mitigate a phenomenon termed *contextual overthinking*. *Contextual overthinking* is mitigated by applying a more context-sensitive and lenient evaluation criterion. Interestingly, both instances of *overthinking* identified in this study appear to stem from the inherent tendency of language models to align their outputs with user prompts. This observation suggests the potential existence of a broader taxonomy of errors, with implications extending to other natural language processing tasks. This experiment highlights the potential of training a reward model to increase the reasoning capabilities of language models and identifies a common taxonomy of improvements that future researchers can focus on.

6.2. Limitations and Assumptions

This section discusses current limitations and assumptions made while designing the experiments.

Noisy evidence

The retrieved evidence is re-ranked based on cosine similarity with the claim using sentence transformer embeddings. These evidence documents are noisy and contain confounding figures and information completely unrelated to the claim. This plays a vital role in the veracity prediction performance. In this study, this is treated as a feature that comes with retrieving evidence from the real world and incorporates this while assessing the reasoning capabilities of the models. To do so, an assumption is made that all the language models used do not exhibit any model-specific bias to certain noise.

Faithful explanation

Another assumption made in these experiments is that the justification produced by the language models to reason decision is faithful to their actual reasoning process. The reasoning is considered valid as long as key evidence pairs pertaining to determine the veracity of the claim is present in the justification document.

Choice of hyperparameter values

Due to resource constraints, experiments were conducted with only one set of hyperparameter values. For adaptive decomposition, the temperature value was set to 0.3 while it is set to 0.45 for test time scaling experiments. This choice allows the model to have some variations in the candidate output sequences generated for training and evaluating the reward models. These values are design choices and are not grounded in literature justifying those choices. No ablation studies were conducted to study the impact of the hyperparameter values.

6.3. Future works

Performance gap with respect to upper-bound

A key observation made is that the proposed methods failed to reach their respective theoretical upper-bound in both the experiments. While it is desirable for the methods to not reach the upper-bound (we do not want our methods to overfit to the dataset), there is a considerable gap in the performance.

For adaptive decomposition, this gap can be bridged with an alternative proxy definition for *Complexity* that aligns better with the nuances of the upper-bound decision. Stronger models to extract latent embedding or a stronger base model for the setfit decision machine can be explored that will be able to capture the training distribution better [107].

In test-time scaling, a stronger reward model and better quality training data can be explored to bridge the performance gap. Better quality training data can be annotated by using human preferences to select better reasoning paths in the output sequences [26].

In-depth study on Overthinking

This study identifies two types of *overthinking* that contribute to overly strict evaluations and reasoning by the language model. It emphasizes the importance of mitigating decomposition-induced overthinking and highlights the need for a subjective assessment of contextual overthinking before any mitigation efforts, as discussed in Sections 5.1 and 5.5. Notably, the issue of overthinking is not confined to fact-checking tasks but extends to a broader range of natural language processing applications [19, 18]. Future research should aim to formalize the definitions of these phenomena and develop task-specific strategies for effective mitigation.

6.4. Disclosure

In line with TU Delft's publishing policies,[1] I acknowledge the use of AI tools powered by ChatGPT to assist in rephrasing certain sections of this thesis. The icons featured in the figures were created by various contributors from `www.flaticon.com`. All AI-generated content has been thoroughly reviewed to ensure its accuracy.

[1] TU Delft publishing policies: `https://www.tudelft.nl/library/actuele-themas/open-publishing/about/policies`

References

[1] Soroush Vosoughi, Deb Roy, and Sinan Aral. “The spread of true and false news online”. In: *Science* 359.6380 (2018), pp. 1146–1151. DOI: 10.1126/science.aap9559. eprint: https://www.science.org/doi/pdf/10.1126/science.aap9559. URL: https://www.science.org/doi/abs/10.1126/science.aap9559.

[2] Kai Shu et al. “Fake news detection on social media: A data mining perspective”. In: *ACM SIGKDD explorations newsletter* 19.1 (2017), pp. 22–36.

[3] Venktesh V et al. *QuanTemp: A real-world open-domain benchmark for fact-checking numerical claims*. 2024. arXiv: 2403.17169 [cs.CL]. URL: https://arxiv.org/abs/2403.17169.

[4] Namika Sagara and Ellen Peters. “Consumer Understanding and Use of Numeric Information in Product Claims”. In: Jan. 2015, pp. 245–245. ISBN: 978-3-319-11796-6. DOI: 10.1007/978-3-319-11797-3_140.

[5] Washington State Office of the Attorney General. *AG Ferguson sues one of the nation’s largest opioid manufacturers over state’s opioid epidemic*. Accessed: 2025-05-12. Sept. 2017. URL: https://www.atg.wa.gov/news/news-releases/ag-ferguson-sues-one-nation-s-largest-opioid-manufacturers-over-state-s-opioid.

[6] Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. “A Survey on Automated Fact-Checking”. In: *Transactions of the Association for Computational Linguistics* 10 (2022). Ed. by Brian Roark and Ani Nenkova, pp. 178–206. DOI: 10.1162/tacl_a_00454. URL: https://aclanthology.org/2022.tacl-1.11.

[7] Andreas Vlachos and Sebastian Riedel. “Fact Checking: Task definition and dataset construction”. In: *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. Ed. by Cristian Danescu-Niculescu-Mizil et al. Baltimore, MD, USA: Association for Computational Linguistics, June 2014, pp. 18–22. DOI: 10.3115/v1/W14-2508. URL: https://aclanthology.org/W14-2508.

[8] Neema Kotonya and Francesca Toni. “Explainable Automated Fact-Checking for Public Health Claims”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber et al. Online, Nov. 2020.

[9] Pepa Atanasova et al. “Generating Fact Checking Explanations”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 7352–7364. DOI: 10.18653/v1/2020.acl-main.656. URL: https://aclanthology.org/2020.acl-main.656.

[10] Wojciech Ostrowski et al. “Multi-Hop Fact Checking of Political Claims”. English. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. Vol. CoRR 2020. arXiv.org. 30th International Joint Conference on Artificial Intelligence ; Conference date: 19-08-2021 Through 27-08-2021. International Joint Conferences on Artificial Intelligence, 2021, pp. 3892–3898. DOI: 10.24963/ijcai.2021/536.

[11] Michael Schlichtkrull, Zhijiang Guo, and Andreas Vlachos. *AVeriTeC: A Dataset for Real-world Claim Verification with Evidence from the Web*. 2023. arXiv: 2305.13117 [cs.CL]. URL: https://arxiv.org/abs/2305.13117.

[12] Angela Fan et al. *Generating Fact Checking Briefs*. 2020. arXiv: 2011.05448 [cs.CL]. URL: https://arxiv.org/abs/2011.05448.

[13] Jifan Chen et al. *Generating Literal and Implied Subquestions to Fact-check Complex Claims*. 2022. arXiv: 2205.06938 [cs.CL]. URL: https://arxiv.org/abs/2205.06938.

[14] Mubashara Akhtar et al. *Exploring the Numerical Reasoning Capabilities of Language Models: A Comprehensive Analysis on Tabular Data*. 2023. arXiv: 2311.02216 [cs.CL]. URL: https://arxiv.org/abs/2311.02216.

[15] Liangming Pan et al. *Fact-Checking Complex Claims with Program-Guided Reasoning*. 2023. arXiv: 2305.12744 [cs.CL]. URL: https://arxiv.org/abs/2305.12744.

[16] Ansh Radhakrishnan et al. *Question Decomposition Improves the Faithfulness of Model-Generated Reasoning*. 2023. arXiv: 2307.11768 [cs.CL]. URL: https://arxiv.org/abs/2307.11768.

[17] Qisheng Hu, Quanyu Long, and Wenya Wang. *Decomposition Dilemmas: Does Claim Decomposition Boost or Burden Fact-Checking Performance?* 2024. arXiv: 2411.02400 [cs.IR]. URL: https://arxiv.org/abs/2411.02400.

[18] Aswin RRV et al. *Chaos with Keywords: Exposing Large Language Models Sycophantic Hallucination to Misleading Keywords and Evaluating Defense Strategies*. 2024. arXiv: 2406.03827 [cs.CL]. URL: https://arxiv.org/abs/2406.03827.

[19] Freda Shi et al. *Large Language Models Can Be Easily Distracted by Irrelevant Context*. 2023. arXiv: 2302.00093 [cs.CL]. URL: https://arxiv.org/abs/2302.00093.

[20] Peiyi Wang et al. *Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations*. 2024. arXiv: 2312.08935 [cs.AI]. URL: https://arxiv.org/abs/2312.08935.

[21] Xuezhi Wang et al. *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. 2023. arXiv: 2203.11171 [cs.CL]. URL: https://arxiv.org/abs/2203.11171.

[22] Shunyu Yao et al. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*. 2023. arXiv: 2305.10601 [cs.CL]. URL: https://arxiv.org/abs/2305.10601.

[23] Charlie Snell et al. *Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters*. 2024. arXiv: 2408.03314 [cs.LG]. URL: https://arxiv.org/abs/2408.03314.

[24] Jonathan Uesato et al. *Solving math word problems with process- and outcome-based feedback*. 2022. arXiv: 2211.14275 [cs.LG]. URL: https://arxiv.org/abs/2211.14275.

[25] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG]. URL: https://arxiv.org/abs/2110.14168.

[26] Hunter Lightman et al. *Let's Verify Step by Step*. 2023. arXiv: 2305.20050 [cs.LG]. URL: https://arxiv.org/abs/2305.20050.

[27] James Thorne et al. "FEVER: a Large-scale Dataset for Fact Extraction and VERification". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Ed. by Marilyn Walker, Heng Ji, and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 809–819. DOI: 10.18653/v1/N18-1074. URL: https://aclanthology.org/N18-1074.

[28] Rami Aly et al. "The Fact Extraction and VERification Over Unstructured and Structured information (FEVEROUS) Shared Task". In: *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*. Ed. by Rami Aly et al. Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1–13. DOI: 10.18653/v1/2021.fever-1.1. URL: https://aclanthology.org/2021.fever-1.1/.

[29] Tal Schuster, Adam Fisch, and Regina Barzilay. *Get Your Vitamin C! Robust Fact Verification with Contrastive Evidence*. 2021. arXiv: 2103.08541 [cs.CL]. URL: https://arxiv.org/abs/2103.08541.

[30] Ryo Kamoi et al. *WiCE: Real-World Entailment for Claims in Wikipedia*. 2023. arXiv: 2303.01432 [cs.CL]. URL: https://arxiv.org/abs/2303.01432.

[31] David Wadden et al. "Fact or Fiction: Verifying Scientific Claims". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 7534–7550. DOI: 10.18653/v1/2020.emnlp-main.609. URL: https://aclanthology.org/2020.emnlp-main.609/.

[32] Julian Eisenschlos et al. "Fool Me Twice: Entailment from Wikipedia Gamification". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by Kristina Toutanova et al. Online: Association for Computational Linguistics, June 2021, pp. 352–365. DOI: 10.18653/v1/2021.naacl-main.32. URL: https://aclanthology.org/2021.naacl-main.32/.

[33] Arkadiy Saakyan, Tuhin Chakrabarty, and Smaranda Muresan. “COVID-Fact: Fact Extraction and Verification of Real-World Claims on COVID-19 Pandemic”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 2116–2129. DOI: 10.18653/v1/2021.acl-long.165. URL: https://aclanthology.org/2021.acl-long.165/.

[34] William Yang Wang. ““Liar, Liar Pants on Fire”: A New Benchmark Dataset for Fake News Detection”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Ed. by Regina Barzilay and Min-Yen Kan. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 422–426. DOI: 10.18653/v1/P17-2067. URL: https://aclanthology.org/P17-2067/.

[35] Tariq Alhindi, Savvas Petridis, and Smaranda Muresan. “Where is Your Evidence: Improving Fact-checking by Justification Modeling”. In: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Ed. by James Thorne et al. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 85–90. DOI: 10.18653/v1/W18-5513. URL: https://aclanthology.org/W18-5513/.

[36] Yichen Jiang et al. “HoVer: A Dataset for Many-Hop Fact Extraction And Claim Verification”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Ed. by Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 3441–3460. DOI: 10.18653/v1/2020.findings-emnlp.309. URL: https://aclanthology.org/2020.findings-emnlp.309.

[37] Zhilin Yang et al. “HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Ed. by Ellen Riloff et al. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2369–2380. DOI: 10.18653/v1/D18-1259. URL: https://aclanthology.org/D18-1259/.

[38] Mourad Sarrouti et al. “Evidence-based Fact-Checking of Health-related Claims”. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Ed. by Marie-Francine Moens et al. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3499–3512. DOI: 10.18653/v1/2021.findings-emnlp.297. URL: https://aclanthology.org/2021.findings-emnlp.297/.

[39] Thomas Diggelmann et al. *CLIMATE-FEVER: A Dataset for Verification of Real-World Climate Claims*. 2021. arXiv: 2012.00614 [cs.CL]. URL: https://arxiv.org/abs/2012.00614.

[40] Ashim Gupta and Vivek Srikumar. “X-Fact: A New Benchmark Dataset for Multilingual Fact Checking”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language*

Processing (Volume 2: Short Papers). Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 675–682. DOI: 10.18653/v1/2021.acl-short.86. URL: https://aclanthology.org/2021.acl-short.86/.

[41] Andreas Hanselowski et al. "A Richly Annotated Corpus for Different Tasks in Automated Fact-Checking". In: *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. Ed. by Mohit Bansal and Aline Villavicencio. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 493–503. DOI: 10.18653/v1/K19-1046. URL: https://aclanthology.org/K19-1046.

[42] Isabelle Augenstein et al. "MultiFC: A Real-World Multi-Domain Dataset for Evidence-Based Fact Checking of Claims". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui et al. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4685–4697. DOI: 10.18653/v1/D19-1475. URL: https://aclanthology.org/D19-1475/.

[43] Mike Lewis et al. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. URL: https://aclanthology.org/2020.acl-main.703.

[44] Nedjma Ousidhoum, Zhangdie Yuan, and Andreas Vlachos. "Varifocal Question Generation for Fact-checking". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 2532–2544. DOI: 10.18653/v1/2022.emnlp-main.163. URL: https://aclanthology.org/2022.emnlp-main.163.

[45] Mark Chen et al. *Evaluating Large Language Models Trained on Code*. 2021. arXiv: 2107.03374 [cs.LG]. URL: https://arxiv.org/abs/2107.03374.

[46] Rami Aly et al. *FEVEROUS: Fact Extraction and VERification Over Unstructured and Structured information*. 2021. arXiv: 2106.05707 [cs.CL]. URL: https://arxiv.org/abs/2106.05707.

[47] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[48] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL]. URL: https://arxiv.org/abs/1907.11692.

[49] Zhengbao Jiang et al. *Active Retrieval Augmented Generation*. 2023. arXiv: 2305.06983 [cs.CL]. URL: https://arxiv.org/abs/2305.06983.

[50] Weihang Su et al. *DRAGIN: Dynamic Retrieval Augmented Generation based on the Information Needs of Large Language Models*. 2024. arXiv: 2403.10081 [cs.CL]. URL: https://arxiv.org/abs/2403.10081.

[51] Xanh Ho et al. "Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Ed. by Donia Scott, Nuria Bel, and Chengqing Zong. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 6609–6625. DOI: 10.18653/v1/2020.coling-main.580. URL: https://aclanthology.org/2020.coling-main.580/.

[52] Mor Geva et al. *Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies*. 2021. arXiv: 2101.02235 [cs.CL]. URL: https://arxiv.org/abs/2101.02235.

[53] James Ferguson et al. *IIRC: A Dataset of Incomplete Information Reading Comprehension Questions*. 2020. arXiv: 2011.07127 [cs.CL]. URL: https://arxiv.org/abs/2011.07127.

[54] Hongru Wang et al. *Self-DC: When to Reason and When to Act? Self Divide-and-Conquer for Compositional Unknown Questions*. 2025. arXiv: 2402.13514 [cs.CL]. URL: https://arxiv.org/abs/2402.13514.

[55] Akari Asai et al. *Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection*. 2023. arXiv: 2310.11511 [cs.CL]. URL: https://arxiv.org/abs/2310.11511.

[56] Soyeong Jeong et al. *Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity*. 2024. arXiv: 2403.14403 [cs.CL]. URL: https://arxiv.org/abs/2403.14403.

[57] Pranav Rajpurkar, Robin Jia, and Percy Liang. *Know What You Don't Know: Unanswerable Questions for SQuAD*. 2018. arXiv: 1806.03822 [cs.CL]. URL: https://arxiv.org/abs/1806.03822.

[58] Tom Kwiatkowski et al. "Natural Questions: A Benchmark for Question Answering Research". In: *Transactions of the Association for Computational Linguistics* 7 (Aug. 2019), pp. 453–466. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00276. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00276/1923288/tacl_a_00276.pdf. URL: https://doi.org/10.1162/tacl%5C_a%5C_00276.

[59] Mandar Joshi et al. "TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Regina Barzilay and Min-Yen Kan. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1601–1611. DOI: 10.18653/v1/P17-1147. URL: https://aclanthology.org/P17-1147/.

[60] Harsh Trivedi et al. "♫ MuSiQue: Multihop Questions via Single-hop Question Composition". In: *Transactions of the Association for Computational Linguistics* 10 (May 2022), pp. 539–554. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00475. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00475/2020694/tacl_a_00475.pdf. URL: https://doi.org/10.1162/tacl%5C_a%5C_00475.

[61] Huanshuo Liu et al. *CtrlA: Adaptive Retrieval-Augmented Generation via Inherent Control*. 2024. arXiv: 2405.18727 [cs.CL]. URL: https://arxiv.org/abs/2405.18727.

[62] Trenton Bricken et al. "Towards Monosemanticity: Decomposing Language Models With Dictionary Learning". In: *Transformer Circuits Thread* (2023). https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[63] Adly Templeton et al. "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet". In: *Transformer Circuits Thread* (2024). URL: https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

[64] Xidong Feng et al. *Alphazero-like Tree-Search can Guide Large Language Model Decoding and Training*. 2024. arXiv: 2309.17179 [cs.LG]. URL: https://arxiv.org/abs/2309.17179.

[65] Yuxi Xie et al. *Self-Evaluation Guided Beam Search for Reasoning*. 2023. arXiv: 2305.00633 [cs.CL]. URL: https://arxiv.org/abs/2305.00633.

[66] DeepSeek-AI et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. 2025. arXiv: 2501.12948 [cs.CL]. URL: https://arxiv.org/abs/2501.12948.

[67] OpenAI. *Learning to Reason with LLMs*. https://openai.com/index/learning-to-reason-with-llms/. Accessed: 2025-04-26. 2024.

[68] Eric Zelikman et al. *Quiet-STaR: Language Models Can Teach Themselves to Think Before Speaking*. 2024. arXiv: 2403.09629 [cs.CL]. URL: https://arxiv.org/abs/2403.09629.

[69] Avi Singh et al. *Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models*. 2024. arXiv: 2312.06585 [cs.LG]. URL: https://arxiv.org/abs/2312.06585.

[70] Yuntao Bai et al. *Constitutional AI: Harmlessness from AI Feedback*. 2022. arXiv: 2212.08073 [cs.CL]. URL: https://arxiv.org/abs/2212.08073.

[71] Aman Madaan et al. "Self-Refine: Iterative Refinement with Self-Feedback". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 46534–46594. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

[72] Eric Zelikman et al. *STaR: Bootstrapping Reasoning With Reasoning*. 2022. arXiv: 2203.14465 [cs.LG]. URL: https://arxiv.org/abs/2203.14465.

[73] Yuxiao Qu et al. *Recursive Introspection: Teaching Language Model Agents How to Self-Improve*. 2024. arXiv: 2407.18219 [cs.LG]. URL: https://arxiv.org/abs/2407.18219.

[74] William Saunders et al. *Self-critiquing models for assisting human evaluators*. 2022. arXiv: 2206.05802 [cs.CL]. URL: https://arxiv.org/abs/2206.05802.

[75] Bowen Jin et al. *Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning*. 2025. arXiv: 2503.09516 [cs.CL]. URL: https://arxiv.org/abs/2503.09516.

[76] Eric Nichols, Leo Gao, and Randy Gomez. *Collaborative Storytelling with Large-scale Neural Language Models*. 2020. arXiv: 2011.10208 [cs.CL]. URL: https://arxiv.org/abs/2011.10208.

[77] Jianhao Shen et al. *Generate Rank: A Multi-task Framework for Math Word Problems*. 2021. arXiv: 2109.03034 [cs.CL]. URL: https://arxiv.org/abs/2109.03034.

[78] Yan Wang, Xiaojiang Liu, and Shuming Shi. "Deep Neural Solver for Math Word Problems". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 845–854. DOI: 10.18653/v1/D17-1088. URL: https://aclanthology.org/D17-1088/.

[79] Rik Koncel-Kedziorski et al. "MAWPS: A Math Word Problem Repository". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by Kevin Knight, Ani Nenkova, and Owen Rambow. San Diego, California: Association for Computational Linguistics, June 2016, pp. 1152–1157. DOI: 10.18653/v1/N16-1136. URL: https://aclanthology.org/N16-1136/.

[80] Dan Hendrycks et al. *Measuring Mathematical Problem Solving With the MATH Dataset*. 2021. arXiv: 2103.03874 [cs.LG]. URL: https://arxiv.org/abs/2103.03874.

[81] Levente Kocsis and Csaba Szepesvári. "Bandit Based Monte-Carlo Planning". In: *Machine Learning: ECML 2006*. Ed. by Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293. ISBN: 978-3-540-46056-5.

[82] Rémi Coulom. "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search". In: vol. 4630. May 2006. ISBN: 978-3-540-75537-1. DOI: 10.1007/978-3-540-75538-8_7.

[83] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. ISSN: 1476-4687. DOI: 10.1038/nature16961. URL: https://doi.org/10.1038/nature16961.

[84] Maciej Świechowski et al. “Monte Carlo Tree Search: a review of recent modifications and applications”. In: *Artificial Intelligence Review* 56.3 (Mar. 2023), pp. 2497–2562. DOI: 10.1007/s10462-022-10228-y. URL: https://doi.org/10.1007/s10462-022-10228-y.

[85] Xidong Feng et al. *Alphazero-like Tree-Search can Guide Large Language Model Decoding and Training*. 2024. arXiv: 2309.17179 [cs.LG]. URL: https://arxiv.org/abs/2309.17179.

[86] Qi Sun et al. *Transformer Layers as Painters*. 2025. arXiv: 2407.09298 [cs.CL]. URL: https://arxiv.org/abs/2407.09298.

[87] Yung-Sung Chuang et al. “DoLa: Decoding by Contrasting Layers Improves Factuality in Large Language Models”. In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=Th6NyL07na.

[88] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[89] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: http://arxiv.org/abs/1908.10084.

[90] Lewis Tunstall et al. *Efficient Few-Shot Learning Without Prompts*. 2022. DOI: 10.48550/ARXIV.2209.11055. URL: https://arxiv.org/abs/2209.11055.

[91] Stephen Robertson and Hugo Zaragoza. “The Probabilistic Relevance Framework: BM25 and Beyond”. In: *Found. Trends Inf. Retr.* 3.4 (Apr. 2009), pp. 333–389. ISSN: 1554-0669. DOI: 10.1561/1500000019. URL: https://doi.org/10.1561/1500000019.

[92] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Levenshtein Distance: Information theory, Computer science, String (computer science), String metric, Damerau?Levenshtein distance, Spell checker, Hamming distance*. Alpha Press, 2009. ISBN: 6130216904.

[93] Lampros Mouselimis. *fuzzywuzzyR: Fuzzy String Matching*. R package version 1.0.5. 2021. URL: https://CRAN.R-project.org/package=fuzzywuzzyR.

[94] SeatGeek Inc. *fuzzywuzzy: Fuzzy String Matching in Python*. 2014. URL: https://github.com/seatgeek/fuzzywuzzy.

[95] Thomas Kluyver et al. “Jupyter Notebooks – a publishing format for reproducible computational workflows”. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press, 2016, pp. 87–90. ISBN: 978-1-61499-649-1. DOI: 10.3233/978-1-61499-649-1-87.

[96] Abhimanyu Dubey et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[97] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL]. URL: https://arxiv.org/abs/1910.03771.

[98] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL]. URL: https://arxiv.org/abs/2310.06825.

[99] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[100] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[101] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL]. URL: https://arxiv.org/abs/2106.09685.

[102] Robert E. White and Karyn Cooper. "Grounded Theory". In: *Qualitative Research in the Post-Modern Era: Critical Approaches and Selected Methodologies*. Cham: Springer International Publishing, 2022, pp. 339–385. ISBN: 978-3-030-85124-8. DOI: 10.1007/978-3-030-85124-8_9. URL: https://doi.org/10.1007/978-3-030-85124-8_9.

[103] Lillemor R-M. Hallberg and. "The "core category" of grounded theory: Making constant comparisons". In: *International Journal of Qualitative Studies on Health and Well-being* 1.3 (2006), pp. 141–148. DOI: 10.1080/17482620600858399. eprint: https://doi.org/10.1080/17482620600858399. URL: https://doi.org/10.1080/17482620600858399.

[104] Sanghyun Park et al. "The Presence of Unexpected Biases in Online Fact-Checking". In: *Harvard Kennedy School (HKS) Misinformation Review* (2021). DOI: 10.37016/mr-2020-53. URL: https://doi.org/10.37016/mr-2020-53.

[105] Won-Ki Moon and Lee Ann Kahlor. "Fact-checking in the age of AI: Reducing biases with non-human information sources". In: *Technology in Society* 80 (2025), p. 102760. ISSN: 0160-791X. DOI: https://doi.org/10.1016/j.techsoc.2024.102760. URL: https://www.sciencedirect.com/science/article/pii/S0160791X24003087.

[106] Michael Soprano et al. "Cognitive Biases in Fact-Checking and Their Countermeasures: A Review". In: *Information Processing Management* 61.3 (2024), p. 103672. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2024.103672. URL: https://www.sciencedirect.com/science/article/pii/S0306457324000323.

[107] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361.

[108] Liyan Tang, Philippe Laban, and Greg Durrett. *MiniCheck: Efficient Fact-Checking of LLMs on Grounding Documents*. 2024. arXiv: 2404.10774 [cs.CL]. URL: https://arxiv.org/abs/2404.10774.

[109] Eurostat. *Electricity price statistics*. Accessed: 2025-05-10. 2025. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Electricity_price_statistics#Explore_further.

A

Appendix

A.1. Definition of the taxonomy of errors of True class when run with decomposition.

Error Taxonomy	Description
Decomposition induced overthinking	• Reasoning over details in evidence not pertinent to predicting the veracity of the claim • Being overly strict with details and figures that could be lenient • Explicitly verifying individual subquestions and poorly aggregating their verdicts to form a final verdict
Noisy Reranked Evidence	• Reranking evidences that contain similar values from similar stories (concerning the subject - person, place, event, etc). • Reranking enitrely noisy evidences, not pertinent to the claim • Certain evidences important for an aspect of the claim not reranked.

Table A.1: Definition of the error taxonomies mentioned in Table 5.2. These definitions are obtained by observing qualitative results following the method described in section 4.6.2

A.2. Definition of the taxonomy of improvements made using Reward Model over majority voting.

Taxonomy of Improvement	Description
Improved reasoning	• Effectively aggregates multiple contextual elements to construct coherent and thorough reasoning. • Comprehend and correctly reason against nuanced numerical, temporal and quantitative values. • Inferring to the correct verdict over the evidence. • Successfully filters out confounding details in the evidence and maintains focused reasoning on claim-relevant content. • Robustly ignores irrelevant or noisy evidence, focusing only on information pertinent to claim verification.
Mitigation of contextual overthinking	• Focusing the analysis on claim-relevant portions of the evidence while ignoring unrelated details present in the evidence. • Reasoning strictly based on the stated claim, without diverging into irrelevant implications or hypothetical extensions.

Table A.2: Definition of taxonomies of improvements made by selecting a reasoning path using a reward model over majority voting. These definitions are obtained by observing qualitative results following the method described in section 4.6.2

A.3. Mathematical formulae required for methodology

A.3.1. Cosine similarity

Cosine similarity is a commonly used metric to measure the similarity between two non-zero vectors based on the angle between them, rather than their magnitude. Given two vectors $\mathbf{x}$ and $\mathbf{y}$, the cosine similarity is defined as:

$$\text{cos\_sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \, \|\mathbf{y}\|}$$

where $\mathbf{x} \cdot \mathbf{y}$ denotes the dot product of the vectors, and $\|\mathbf{x}\|$ and $\|\mathbf{y}\|$ represent their Euclidean norms.

Cosine similarity ranges from -1 to 1, where:

- 1 indicates that the vectors point in the same direction (maximum similarity),
- 0 means the vectors are orthogonal (no similarity),
- -1 means the vectors are diametrically opposed.

In the context of this work, cosine similarity is used to compare the latent representations of tokens, layers, or classes in the embedding space, providing an interpretable metric of semantic similarity.

A.3.2. F1 Score and Averaging Methods

The F1 score is a commonly used evaluation metric in classification tasks, particularly in scenarios where the class distribution is imbalanced. It is defined as the harmonic mean of precision and recall, and it provides a single measure that balances the trade-off between these two quantities.

F1 Score Definition

Given a classification task, let TP be the number of true positives, FP the number of false positives, and FN the number of false negatives. Then, precision and recall are defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{A.1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{A.2}$$

The F1 score is calculated as the harmonic mean of precision and recall:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{A.3}$$

The F1 score ranges from 0 to 1, with higher values indicating better performance. It is especially useful in situations where both false positives and false negatives carry significant consequences, and where class imbalance may render accuracy an unreliable metric.

Macro and Weighted Averaging

For multi-class classification problems, F1 scores are often averaged across classes to provide a summary measure. Two common strategies are macro averaging and weighted averaging.

Macro-Averaged F1 Score

Macro averaging computes the F1 score independently for each class and then takes the unweighted mean. This treats all classes equally, regardless of their support (i.e., the number of true instances per class).

$$\mathsf{F1}_{\mathsf{macro}} = \frac{1}{C} \sum_{i=1}^{C} \mathsf{F1}_i \tag{A.4}$$

where C is the total number of classes and $\mathsf{F1}_i$ is the F1 score for class i.

Weighted-Averaged F1 Score

Weighted averaging also computes the F1 score for each class, but takes the mean weighted by the number of true instances in each class. This ensures that the metric reflects the performance on more frequent classes proportionally.

$$\mathsf{F1}_{\mathsf{weighted}} = \sum_{i=1}^{C} \left(\frac{n_i}{N} \cdot \mathsf{F1}_i \right) \tag{A.5}$$

where n_i is the number of true instances of class i, $N = \sum_{i=1}^{C} n_i$ is the total number of instances, and $\mathsf{F1}_i$ is the F1 score for class i.

These averaging strategies are essential for understanding classifier performance in imbalanced datasets and help provide a more comprehensive evaluation beyond simple accuracy.

A.4. Detailed algorithm to assign taxonomy of errors/improvements using qualitative analysis.

Algorithm 2 Iterative Construction of an Error Taxonomy

Require: Initial LLM outputs D_0
Ensure: Final taxonomy of error categories $\mathcal{C}$

Initialize pattern set, code set, and category set
$D \leftarrow D_0$
while taxonomy is not theoretically saturated **do**
    for each output $d_i \in D$ **do**
        Identify reasoning patterns in d_i
        Compare patterns across and within samples
        Write memos capturing analytical insights
        Assign descriptive codes to distinct patterns ▷ Open coding
    end for
    Group codes into higher-level categories ▷ Axial coding
    Refine category definitions based on relationships between codes
    Collect additional LLM outputs D_{new} and append to D
    for each new pattern $p \in D_{\text{new}}$ **do**
        if p contradicts an existing category **then**
            Redefine or split the category using steps 3 - 12
        else if p expands an existing category **then**
            Update the category to include new dimension using steps 3 - 12
        else
            Mark the category as supported
        end if
    end for
    Check if all categories are supported ▷ Exit condition for while loop
    if all categories are supported **then**
        Set taxonomy to be theoretically saturated
    end if
end while
Return the final set of error categories $\mathcal{C}$

A.5. System prompt and user prompt for veracity prediction inference model

You are a fact-checking assistant tasked with evaluating claims based on provided evidence. Your task:

1. Carefully analyze the important details from both the claim and the evidence provided.

2. Reason through the evidence step-by-step, synthesizing relevant information to assess the overall veracity of the claim.

3. Finally Classify the claim's veracity into one of the following categories:
   (a) True: The evidence fully supports the claim.
   (b) False: The evidence directly contradicts the claim.
   (c) Conflicting: The evidence contains contradictory elements or is inconclusive.

Response Format:
- [Label]: (True, False, or Conflicting)
- [Justification]: The reasoning steps used that led to your classification of the claim.

Figure A.1: System prompt to veracity prediction inference model for *without decomposition* inference.

You are a fact-checking assistant tasked with evaluating claims based on provided evidence and guiding subquestions. Each claim is accompanied by subquestions, which are designed to help you focus on key aspects of the claim in relation to the evidence provided. Use these subquestions as a guide to structure your analysis but **do not independently classify the subquestions**. Instead, use the insights gained from addressing each subquestion to assess the overall veracity of the main claim. Your task:

1. Carefully analyze the important details from both the claim and the evidence provided, guided by the sub-questions.

2. Reason through the evidence step-by-step, synthesizing relevant information to assess the overall veracity of the claim.

3. Finally Classify the claim's veracity into one of the following categories:
   (a) True: The evidence fully supports the claim.
   (b) False: The evidence directly contradicts the claim.
   (c) Conflicting: The evidence contains contradictory elements or is inconclusive.

Response Format:
- [Label]: (True, False, or Conflicting)
- [Justification]: The reasoning steps used that led to your classification of the claim.

Figure A.2: System prompt to veracity prediction inference model for *with decomposition* inference.

Here are the examples of fact-checking:

1. **[Claim]**: <Example claim 1>
[Sub-questions]: <Example sub-questions 1>
[Evidences]: <Example evidence 1>
[Label]: Conflicting

2. **[Claim]**: <Example claim 2>
[Sub-questions]: <Example sub-questions 2>
[Evidences]: <Example evidence 2>
[Label]: False

Similar to the given examples, fact check the following claim using the evidence. Pay additional attention to numerical spans in claim and evidence and fact check by thinking step by step and output the label by performing entailment. Classify the entire claim strictly into one of the following categories: TRUE, FALSE or CONFLICTING.

[Claim]: <Inference Claim>
[Sub-questions]: <Inference sub-questions>
[Evidences]: <Inference Evidence>

Give the reply in the following format:
[Label]: (SUPPORTS, REFUTES or CONFLICTING).
[Justification]:

Figure A.3: User prompt to veracity prediction inference model. The sub-questions are removed from the inference sample and the examples when run in *without decomposition* setting.

A.6. Additional results - Latency and Compute costs of adaptive decomposition experiments

To provide a comprehensive assessment of the computational efficiency and cost associated with the adaptive decomposition experiments, both latency and cost per claim were evaluated. Latency is reported as the mean and standard deviation across all claims in the test set of `Quantemp` [3], reflecting the average time required to process a single claim under each experimental setting. As inference was conducted on a local server, cost estimation is presented in terms of GPU runtime expenditure rather than cloud compute billing.

Following the approach adopted in the MiniCheck paper [108], the total cost is estimated using the formula:

$$\text{Cost} = \text{Total runtime (in hours)} \times \text{GPU rate per hour} \tag{A.6}$$

In this context, the total runtime denotes the time taken to process the entire `Quantemp` test set, measured in hours. The GPU rate per hour for the NVIDIA GeForce RTX 3090 used in our local setup is calculated to be $0.154. This rate is derived by multiplying the device's power consumption in kilowatt-hours[1] by the average electricity cost in Germany, where the workstation is hosted [109].

The results are summarized in Table A.3. Among all configurations, the claim decomposition setting incurs the highest latency and per-claim cost, whereas the baseline setting without decomposition demonstrates the lowest computational demand. Adaptive decomposition achieves a balance, reducing both latency and cost compared to full decomposition by selectively applying decomposition and avoiding unnecessary overhead from sub-question generation and re-ranking. These results indicate that adaptive decomposition not only enhances prediction performance but also offers improved cost-effectiveness and computational efficiency.

Method	**Latency (ms)**		**Costs ($)**
	Mean	**Std dev**	
Vanilla inferencing			
without Decomposition	7398	2200	0.796
with Decomposition	10335	2386	1.415
Adaptive Decomposition - Latent representation based classifier			
Llama feedback	7980	2283	0.874
gpt feedback	8015	2712	0.882
Adaptive Decomposition - setfit classifier			
Llama feedback	8624	2779	0.981
gpt feedback	8713	2873	1.05

Table A.3: Latency and per claim cost for running decomposition and adaptive decomposition experiments with `llama-3.1-8b-Instruct` model.

[1] `https://www.nvidia.com/en-eu/geforce/graphics-cards/30-series/rtx-3090-3090ti/`