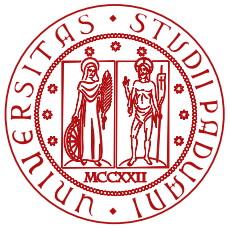


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Department of Information Engineering

Master Thesis in Computer Engineering

Knowledge Graph Fact Verification using Retrieval-Augmented Generation

Master Candidate

Farzad Shami

Student ID 2090160

Supervisor

Prof. Gianmaria Silvello

University of Padova

Co-supervisor

Prof. Stefano Marchesin

University of Padova

Academic Year
2024/2025

Date: nth March 2025

To all those who have believed in me
and encouraged me to pursue my passions.

Abstract

Ensuring the truthfulness of knowledge graphs is critical as they serve as foundational tools in powering many AI systems, search engines, and decision-support systems. This thesis proposes a novel approach using retrieval-augmented generation (RAG) to verify facts in knowledge graphs. The system combines large language models, information retrieval techniques, and a multi-stage verification process to evaluate the veracity of knowledge graph facts. Key components include generating queries from knowledge graph triples, integrating web search for external evidence retrieval, employing embedding-based document chunking and retrieval, and utilizing an ensemble of language models for fact verification. The system aggregates outputs from multiple models using adaptive dispute resolution techniques and majority voting. Comprehensive experiments evaluate various text chunking techniques, embedding models, and document selection procedures. Results demonstrate both the effectiveness of the proposed approach in verifying knowledge graph facts and areas for further optimization. This work advances the evolving fields of knowledge base curation and automated fact checking.

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

CSV Comma Separated Values

NLI Natural Language Inference

NLP Natural Language Processing

LLMs Large Language Models

LLM Large Language Model

QA Question Answering

RAG Retrieval-Augmented Generation

ML Machine Learning

IR Information Retrieval

HTML HyperText Markup Language

RoPE rotary position embeddings

MTEB Massive Text Embedding Benchmark

LoRA Low-Rank Adaptation

MRL Matryoshka Representation Learning

RLHF Reinforcement Learning with Human Feedback

SFT Supervised Fine-Tuning

GQA Grouped-Query Attention

SWA Sliding Window Attention

RS Rejection Sampling

DPO Direct Preference Optimization

1 Introduction

With the rise of big data and AI, it's crucial to have the ability to verify facts and analyze information. Knowledge graphs, which represent knowledge through entities and their relationships, have emerged as an effective tool for arranging and querying massive amounts of structured data. Maintaining the accuracy and dependability of knowledge graphs is a significant challenge. This thesis presents a novel technique for checking facts in knowledge graphs using retrieval-augmented generation, which combines the benefits of Information Retrieval (IR), Natural Language Processing (NLP), and Machine Learning (ML).

1.1 Background and Motivation

A lot of different kinds of apps use knowledge graphs now, from search engines and recommendation systems to question-answering sites and virtual helpers. They store information in a structured way that makes it easy to question and draw conclusions. But current knowledge graphs are so big and complicated that it's hard to check every fact they contain by hand. Because of this, automated fact-checking systems are needed to keep these knowledge sources legitimate and reliable.

Rule-based systems or simple statistical methods are often used in traditional ways to check facts. These methods can work for some types of facts, but they don't work well for more complicated or subtle data. Recent progress in ML and NLP has made it possible for fact checking systems to become smarter.

Large Language Models (LLMs) have shown amazing skills in understanding and producing text that sounds like it was written by a person. This makes them very likely to be successful in tasks that need to verify facts. On the other hand, LLMs have some problems. They can sometimes make up information that sounds reasonable but isn't true, This is called "hallucination." Also, the data they were taught on is all they know, and that data may become out-of-date over time. To get around these problems, this research has come up with retrieval-augmented generation (RAG) methods that mix the best parts of LLMs with information from outside sources.

1.2 Problem Statement

This thesis tackles the issue of automated fact verification in knowledge graphs with a RAG-based methodology. Our objective is to create a system capable of:

- Retrieve relevant information from external sources to support or refute claims in a knowledge graph.
- Utilize LLMs to reason about the retrieved information and generate accurate assessments of fact truthfulness.
- Handle a wide range of fact types and domains, from simple statements to more complex relational facts.
- Provide multiple responses for its verification decisions, enhancing transparency and trust in the system.

1.3 Proposed Approach

Our proposed approach combines several key components to create a robust fact verification system:

- **Knowledge Graph Representation:** We start by representing facts from the knowledge graph in a format suitable for processing by language models and IR systems. This involves converting the subject-predicate-object triples of the knowledge graph into natural language statements.
- **Query Generation:** For each fact to be verified, we generate multiple queries designed to retrieve relevant information from external sources. These queries are formulated to capture different aspects of the fact and potential supporting or contradicting evidence.

- **IR:** We use advanced IR techniques to search for relevant documents or passages from a large corpus of trusted sources. This step leverages both traditional search algorithms and dense retrieval methods based on neural networks.

- **Context Processing:** The retrieved information is processed and combined to create a comprehensive context for each fact. This may involve techniques such as text summarization, entity linking, and coreference resolution to create a coherent representation of the relevant information.

- **Large Language Model (LLM) Integration:** We use several LLMs at the same time to look at the context that was retrieved and decide if the original fact is true. By putting together the results of several models, we hope to reduce the flaws in each one and make the whole thing more accurate.

- **Fact Verification Decision:** The system makes a final decision on the truthfulness of the fact based on the consensus of the language models and the strength of the supporting or contradicting evidence. This decision is accompanied by the reasoning process and relevant evidence.

1.4 Contributions

This thesis makes several key contributions to the field of knowledge graph fact verification:

- A novel pipeline for fact verification that integrates state-of-the-art techniques in IR, NLP, ML.

- A comprehensive evaluation of different retrieval methods, embedding techniques, and language models for the task of fact verification.

- New strategies for generating effective queries and processing retrieved information to support fact verification.

- Analysis of the advantages and drawbacks of employing LLMs for reasoning regarding factual knowledge.

- A detailed analysis of the system's performance across different types of facts and knowledge domains.

1.5 Thesis Structure

The remainder of this thesis is organized as follows:

Chapter 2 provides a comprehensive review of the related works in fact verification, IR, and language model applications through LLMs. It situates our work within the broader context of these research areas and highlights the gaps that our approach aims to address.

Chapter 3 presents a detailed description of our proposed pipeline for fact verification. It explains each component of the system, including the rationale behind design choices and implementation details.

Chapter 4 describes the experimental setup used to evaluate our system. This includes details on the datasets used, evaluation metrics, and baseline systems for comparison and offers an in-depth discussion of the results, exploring the implications of our findings and their potential impact on the field of knowledge graph fact verification.

Chapter 5 Presents a study that investigates the impact of various pipeline components on the system's overall performance, while also exploring different methodologies for each component to determine the optimal final pipeline configuration. Finally, chapter 6 concludes the thesis by summarizing the key contributions, discussing limitations of the current approach, and outlining promising directions for future research.

1.6 Significance and Potential Applications

The development of effective fact verification systems for knowledge graphs has far-reaching implications across various domains:

- **Information Integrity:** Our solution facilitates the automatic verification of facts, hence enhancing the correctness and dependability of extensive knowledge bases. This is especially crucial in a time when disinformation may disseminate swiftly online.
- **Decision Support:** In sectors such as healthcare, finance, and law, where judgments frequently depend on factual information, our technology could function as an essential instrument for validating crucial data points.
- **Educational Applications:** Fact verification systems can be used in educational settings to help students critically evaluate information and develop digital literacy skills.

- **Content Moderation:** Social media platforms and content aggregators may employ similar techniques to detect and flag potentially inaccurate or misleading information.
- **Scientific Research:** In the scientific community, our approach could assist in fact-checking research claims, cross-referencing findings, and identifying potential inconsistencies in the literature.

By addressing the challenge of knowledge graph fact verification, this thesis aims to contribute to the broader goal of creating more reliable and trustworthy information systems. As the volume and complexity of digital information continue to grow, the need for sophisticated fact verification tools becomes increasingly critical. Our work represents a step towards meeting this need, combining the latest advances in artificial intelligence with rigorous IR techniques. In the following chapters, we will delve into the technical details of our approach, present our findings, and explore the implications of this research for the future of knowledge management and information verification.

2

Related Works

This thesis builds upon prior research in knowledge graph fact verification, LLMs, Retrieval-Augmented Generation (RAG), and entailment verification. In this section, we provide an overview of the relevant literature across these areas.

2.1 Entailment Verification and Language Models

In the paper "Minds versus Machines: Rethinking Entailment Verification with Language Models", Sanyal et al. [27] evaluate and compare the inference capabilities of humans and LLMs through a carefully constructed entailment verification benchmark. Their study spans three categories: Natural Language Inference (NLI), contextual Question Answering (QA), and rationales, using multi-sentence premises and diverse types of knowledge to assess inference across complex reasoning scenarios.

The authors found that LLMs generally excel in multi-hop reasoning tasks, particularly those requiring inference over extended contexts, while humans outperform LLMs in simpler deductive reasoning tasks involving substitutions or negations.

Interestingly, both perform comparably in situations requiring inference of missing knowledge. One of the paper's key contributions is the fine-tuning of the Flan-T5 [2] model, which outperforms GPT-3.5 and performs at a comparable level to GPT-4, thus providing a robust, open-source solution for entailment verification tasks. In contrast, the proposed approach to factulizing the

Figure 2.1: Distinctions between human and LLM Inferences. The entailment prediction performance of humans and LLMs are depicted by a 5-star rating scale [27].

knowledge graph using RAG emphasizes the integration of external knowledge retrieval to ground factual assertions, which is critical for generating verifiable, accurate knowledge graphs. While Sanyal et al. focus on the entailment between premises and hypotheses in textual inference, my work extends this by incorporating external evidence to ensure not just consistency but also factual correctness.

In comparison, the entailment verification tasks handled by Sanyal et al. emphasize reasoning within the constraints of the given context, whereas my RAG-based approach highlights the necessity of retrieval from large external datasets to mitigate hallucinations and improve the factual grounding of generated content. Both approaches deal with inference verification but diverge in their method of contextualizing and validating knowledge, with mine incorporating real-time retrieval for fact-checking.

This distinction is significant in terms of application: while their fine-tuned Flan-T5 model achieves high accuracy in entailment tasks, it remains bound to the contextual limits of its training data. My work, by integrating retrieval,

potentially overcomes this limitation by dynamically accessing external data, thus offering a complementary perspective to entailment verification focused on enhancing factuality.

2.2 Claim Verification in the Age of Large Language Models

Dmonte et al. [5] provide a comprehensive survey of LLM-based approaches to claim verification, highlighting the shift from traditional NLP methods to more sophisticated LLM-driven techniques. The typical LLM-based claim verification pipeline, as described by Dmonte et al., consists of several key components:

1. **Evidence Retrieval:** Utilizing techniques like RAG to fetch relevant information from external sources.
2. **Prompt Creation:** Developing effective prompting strategies to guide LLMs in processing claims and evidence.
3. **Transfer Learning:** Employing fine-tuning and in-context learning to adapt LLMs to the specific task of claim verification.
4. **LLM Generation:** Using LLMs to generate veracity labels, supporting evidence, and explanations.

This pipeline represents a departure from traditional fact-checking approaches, leveraging the power of LLMs to improve accuracy and provide more nuanced assessments of claim veracity. Based on survey, several studies have demonstrated the effectiveness of LLM-based approaches in claim verification:

- Zhang and Gao [41] introduced the Hierarchical Step-by-Step (HiSS) prompting method, which directs LLMs to separate a claim into several sub-claims and then verify each via multiple questions-answering steps progressively, improving performance on complex news claim verification tasks.
- Lee et al. [15] developed FactualityPrompts, a framework for assessing the factual accuracy of LLM-generated content.

These studies consistently show that LLM-based methods outperform traditional NLP approaches in terms of accuracy, flexibility, and the ability to handle complex claims.

Our approach shares similarities with the LLM-based pipeline described by Dmonte et al., particularly in the use of retrieval-augmented generation and

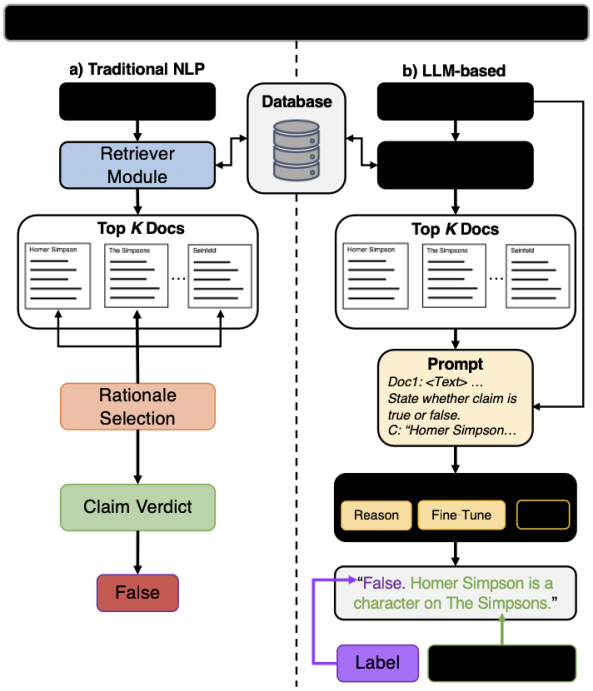


Figure 2.2: Comparison of claim verification systems between NLP-based (traditional) and LLM-based for claim veracity. [5].

the integration of multiple LLMs. However, our method differs in several key aspects:

1. **Multi-Query Retrieval:** We employ a multi-query strategy for evidence retrieval, potentially improving the coverage and relevance of supporting information.
2. **Iterative Refinement:** Our system incorporates an iterative process for refining retrieved evidence and generated responses, which is not explicitly mentioned in most LLM-based approaches surveyed.
3. **Limited Explanation:** While many LLM approaches provide explanations, our method places a stronger emphasis on generating binary (pass/fail) labels for claims to reduce the costs of using LLMs.
4. **Diverse LLM Model:** We use multiple LLMs with diverse architectures to provide more reliable verification result.

These distinctions position our work as a novel contribution to the field, building upon the foundations of LLM-based claim verification while introducing innovative techniques to enhance performance and interpretability.

Despite the promising results of LLM-based claim verification, several challenges remain. Dmonte et al. highlight issues such as handling irrelevant

context, resolving knowledge conflicts, and expanding to multilingual settings. Our approach attempts to address some of these challenges, particularly in the areas of context relevance and explainability. However, there is still significant room for improvement in creating more robust, reliable, and universally applicable claim verification systems.

2.3 Retrieval-Augmented Fact Verification by Synthesizing Contrastive Arguments

The paper Retrieval-Augmented Fact Verification by Synthesizing Contrastive Arguments [38] explores a method for improving fact verification in knowledge graphs using RAG. The proposed framework combines retrieval of external information and the generation of contrastive arguments-claims supported by retrieved evidence, but also those that provide counterpoints. This dual synthesis provides a richer and more nuanced verification process, allowing the system to handle conflicting evidence more effectively. The core contribution of the work lies in the creation of contrastive arguments, a strategy designed to reduce errors in fact verification systems, especially when LLMs may hallucinate or generate incomplete reasoning.

The authors leverage a multi-stage pipeline where external documents are retrieved to support or refute a given claim. Each retrieved piece of evidence is evaluated using a neural network model that ranks the evidence based on its relevance to the claim. By synthesizing contrastive arguments, the system generates explanations for both supporting and refuting the claim, which helps improve the transparency and trustworthiness of the model's decisions.

In terms of results, the framework shows improvement over traditional fact verification pipelines, particularly in handling ambiguous or conflicting information. The contrastive arguments allow for better handling of cases where facts are not binary but exist in a more complex, nuanced state. The system's ability to generate arguments for both sides of a claim increases its robustness and provides a more reliable fact verification tool.

The described approach and my work on fact verification in knowledge graphs using RAG share a common goal: improving the factual accuracy of information through the integration of external knowledge retrieval. However, there are key differences in the methodologies used. The contrastive argument

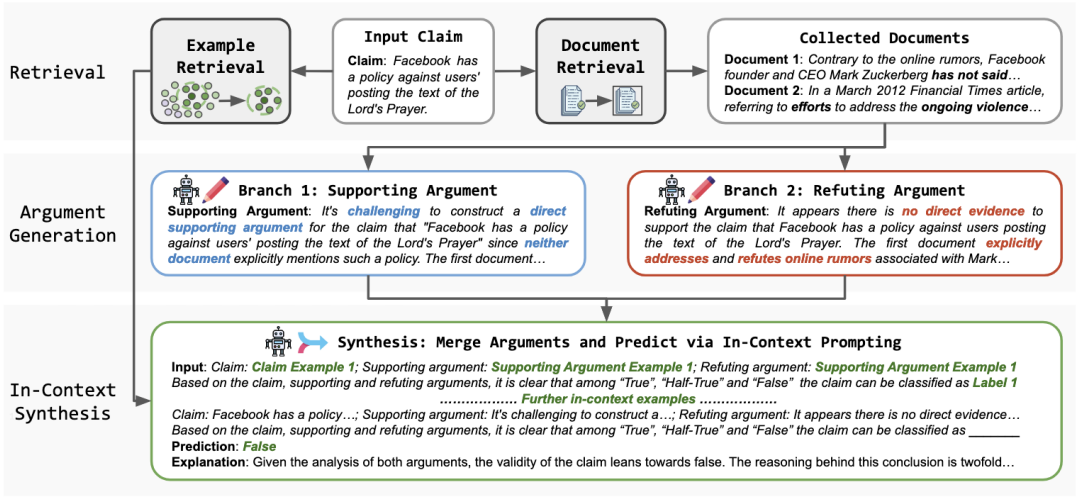


Figure 2.3: The proposed RAFTS [38], which performs few-shot fact verification by incorporating informative in-context demonstrations and contrastive arguments with nuanced information derived from the retrieved documents

synthesis introduced by the authors focuses heavily on generating both supporting and opposing arguments for claims, which provides a more holistic perspective in scenarios where evidence is mixed. In contrast, my approach emphasizes majority voting among multiple models and a multi-query strategy to retrieve a broader range of external evidence, aiming to reduce the incidence of hallucinations in LLM outputs.

While both approaches use retrieval to mitigate the limitations of LLMs, my work incorporates adaptive dispute resolution techniques and focuses on synthesizing outputs from multiple LLMs rather than generating contrastive arguments. This means that my approach leans more towards optimizing model diversity and utilizing the best consensus from several LLMs to ensure factual accuracy, rather than explicitly generating opposing arguments for each claim.

2.4 RAGAR: RAG-Augmented Reasoning for Political Fact-Checking using Multimodal LLMs

The study titled RAGAR: RAG-Augmented Reasoning for Political Fact-Checking using Multimodal LLMs [13] introduces a novel approach to political fact-checking by leveraging RAG with multimodal LLMs. This work focuses on enhancing fact verification in the politically sensitive domain, where disin-

formation can have far-reaching consequences. The authors integrate various modalities text, images, and other media sources into a unified fact-checking pipeline powered by LLMs, particularly emphasizing RAG's ability to retrieve and synthesize external evidence to validate or refute claims.

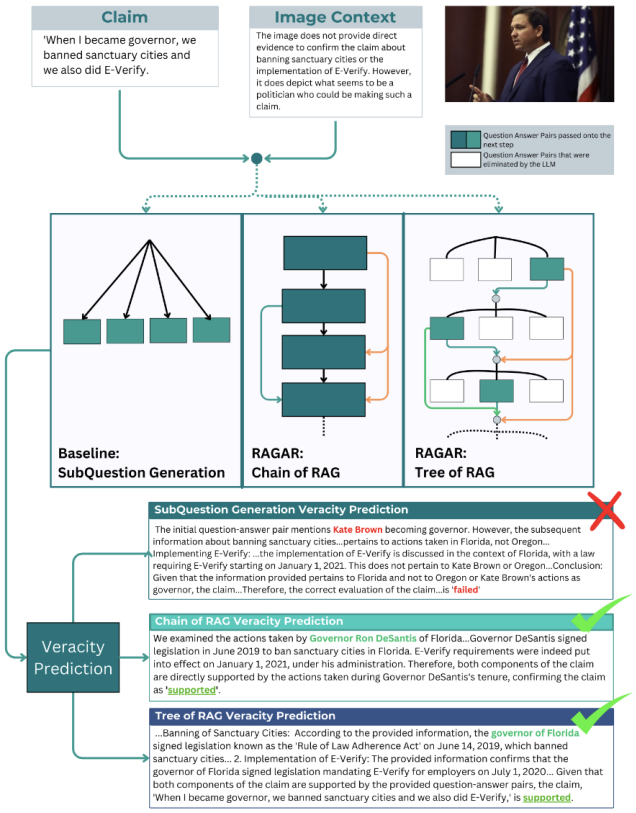


Figure 2.4: An overview of the fact-checking pipeline contrasting the baseline Sub-Question Generation approach from the Chain of RAG and Tree of RAG approach followed by veracity prediction and explanation.

The central innovation of RAGAR lies in its multimodal reasoning capabilities, which allow the model to handle political claims that involve not only textual content but also visual data, such as images or charts. By extending RAG to this multimodal context, the system improves its ability to assess the veracity of claims in real-time, leveraging external resources such as political databases and live web content. Furthermore, the use of contrastive learning helps the system generate both supporting and opposing arguments for each claim, providing a more balanced and comprehensive fact-checking process.

Results from the paper show significant improvements in fact-checking accuracy, particularly for politically charged claims that are often more nuanced

or context-dependent. RAGAR's ability to synthesize multimodal evidence into a coherent verification report highlights its potential for real-world applications, especially in environments where disinformation spreads quickly, such as social media platforms.

RAGAR focuses on political fact-checking using multimodal data, whereas my work targets the factualization of knowledge graphs with a primary focus on textual information. In contrast to RAGAR's multimodal pipeline, my system emphasizes multi-query strategies and document chunking techniques to retrieve highly relevant textual evidence for verification.

3

Pipeline

This chapter outlines a multi-stage NLP pipeline developed specifically for fact verification. The pipeline integrates advanced technologies and methodologies to address core challenges in verifying information accuracy, interpreting user queries, retrieving relevant data from large datasets, and generating factually consistent responses.

The pipeline's foundation is a knowledge graph dataset, which captures complex relationships between entities to support in-depth retrieval and verification. The pipeline applies query generation, cross-encoding for relevance, and multi-layered retrieval strategies to effectively analyze and cross-check information against this structured data.

To access diverse and up-to-date sources, the pipeline incorporates Google Search, blending structured knowledge with real-time information retrieval to enhance verification accuracy.

A key feature of the pipeline is its approach to context processing and information synthesis. By dividing retrieved data into manageable sections and utilizing embedding techniques, the pipeline performs precise analyses, enabling a robust assessment of facts with the support of multiple LLMs, and ensuring thorough verification.

The decision-making framework in the pipeline leverages majority voting across models and employs a final judge to resolve inconsistencies, creating a balanced and coherent output. This approach helps reduce errors and manage potential biases, strengthening the reliability of verified information.

As each pipeline component is further explored, critical insights into its

strengths and limitations are addressed. This includes ethical considerations in AI-driven fact verification, potential biases in knowledge representation, and the broader impacts of advanced verification systems.

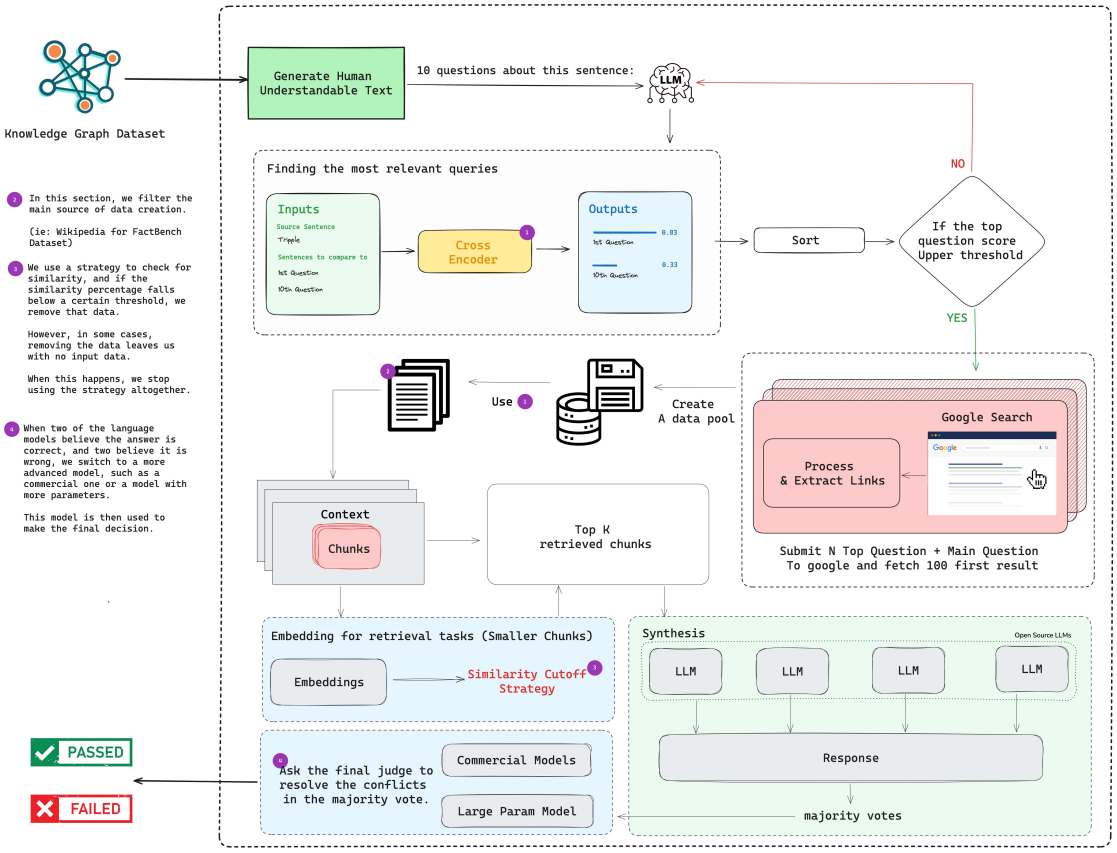


Figure 3.1: RAG-Based Fact Verification Pipeline

This chapter aims to provide a comprehensive analysis of each stage of the pipeline as showed on Figure 3.1, examining not only the technical aspects of its implementation but also the theoretical underpinnings and practical implications of its design choices. By understanding the intricacies of this system, we can gain valuable insights into the current state of NLP technologies and the potential future directions for research and development in this rapidly advancing field. As we proceed, we will explore each component in detail, starting with the foundational knowledge graph dataset and progressing through the various stages of query processing, information retrieval, context analysis, and response generation. This exploration will shed light on the complex interplay between different AI technologies and methodologies, offering a holistic view of how modern NLP systems can be architected to tackle some of the most challenging problems in information processing and human-computer interaction.

3.1 Knowledge Graph Dataset

The foundation of our multi-stage RAG pipeline is the Knowledge Graph Dataset, which acts as the primary source of factual information for the ensuing processing stages. This section clarifies the attributes and aims of the knowledge graph and its use in our pipeline.

3.1.1 Definition and Purpose

In the form of a graph, a knowledge graph is an ordered way to show information that models real-world things and how they relate to each other. The Knowledge Graph Dataset is a huge collection of facts, ideas, and connections that are all linked together in our process. Its main job is to give a lot of background information so that complicated queries can be understood and processed.

The implementation of a knowledge graph fulfills multiple essential functions:

- **Semantic Representation:** Unlike traditional relational databases, knowledge graphs capture semantic relationships between entities, allowing for more nuanced and context-aware information retrieval.
- **Inferential Capabilities:** The interconnected nature of the graph enables the system to make inferences and connections that may not be explicitly stated, enhancing the depth and breadth of responses.
- **Scalability:** Knowledge graphs can efficiently handle large volumes of heterogeneous data, making them ideal for systems that need to process diverse types of information.
- **Flexibility:** The graph structure allows for easy updates and expansions, ensuring that the knowledge base can evolve with new information and changing requirements.

3.1.2 Structure and Components

The Knowledge Graph Dataset in our pipeline is composed of several key components:

- **Nodes:** Representing entities or concepts, nodes are the fundamental units of information in the graph. Each node typically corresponds to a distinct piece of knowledge, such as a person, place, event, or abstract concept.

- **Edges:** These are the connections between nodes, representing relationships or interactions. Edges are often directional and labeled to indicate the nature of the relationship (, "is_a", "part_of", "created_by").
- **Properties:** Nodes and edges can have associated properties or attributes that provide additional details or metadata about the entity or relationship.

3.1.3 Role in the Overall Pipeline

As illustrated in the pipeline diagram, the Knowledge Graph Dataset is the thing that we want to verify the correctness of it. The pipeline uses the knowledge graph to generate queries, retrieve relevant information, and synthesize responses to find the correctness of the knowledge graph.

3.2 Query Generation and Processing

The Query Generation and Processing step, following to the basic Knowledge Graph Dataset, is a pivotal point in our pipeline, wherein user inputs are converted into structured queries suitable for efficient processing by later components. This section clarifies the techniques and methodologies utilized in this critical phase for subsequent actions in the information retrieval process.

3.2.1 Human-Understandable Text Generation

In the first step of the pipeline, we use a LLM (*i.e.* LLama3 [6], Gemma2 [31]) to easily generate human-readable text by submitting prompts. This approach bridges the gap between representing raw data and conveying it in natural language, making the information more accessible and understandable. For additional information, consult the prompt template in Appendix A.1 to observe the sentence generation process.

Key aspects of this process include:

- **Contextual Awareness:** Integrating relevant context from the Knowledge Graph to guarantee that the output content is relevant and useful.
- **Adaptability:** Customizing the generated text to accommodate various complexity levels, catering to diverse user needs and query types.
- **Semantic Enrichment:** Enhancing the generated text with semantic annotations to facilitate more accurate downstream processing.

Be aware that certain knowledge graph datasets are human-readable, whereas others are not; for instance, the FactBench [8] dataset may be easily comprehended by concatenating the subject, predicate, and object of each triple.

Table 3.1: Examples of Human-Understandable Text Generation, illustrates entries from multiple knowledge graphs, detailing the transformation of raw triples into readable sentences.

Knowledge Graph			**Source**	**Is Generated ?**	**Final Text**
Subject	Predicate	Object			
Albert Einstein	*Birth Place*	*Ulm, Germany*	FactBench [8]	✗	*Albert Einstein birth place Ulm, Germany*
Chris Benoit	*deathPlace*	*Fayetteville, Georgia*	FactBench [8]	✗	*Chris Benoit death place Fayetteville, Georgia*
Alexander_III _of_Russia	*isMarriedTo*	*Maria_Feodorovna __Dagmar_of_Denmark_*	YAGO [30]	✓	*Alexander III of Russia is married to Maria Feodorovna, also known as Dagmar of Denmark.*
Shock_to_the _System _(Gemma_Hayes _song)	*length*	*221.0*	DBpedia	✓	*The length of Shock to the System (Gemma Hayes song) is 221.0.*
Paora_Winitana	*years*	*2011*	DBpedia	✓	*Paora Winitana was active in 2011.*

3.2.2 Question Formulation Techniques

A cornerstone of our pipeline is its ability to generate 10 questions about the input sentence, as illustrated in the diagram.

This multi-question approach serves several purposes:

- **Comprehensive Coverage:** By generating multiple questions, the system ensures a thorough exploration of the input's various aspects and potential interpretations.
- **Disambiguation:** Multiple questions help in clarifying ambiguities that may be present in the original input.
- **Context Expansion:** Each generated question potentially introduces new contextual elements, broadening the scope of the subsequent information retrieval process.
- **Robustness:** The diversity of questions increases the likelihood of capturing the user's true intent, even if the original input is vague or imprecise.

Implementing this technique involves using LLMs to generate 10 questions about the input sentence, leveraging the models' language understanding capabilities to ensure the questions are relevant and contextually appropriate. The prompt template used to guide the question generation process is reported in Appendix A.2.

3.2.3 Cross-Encoder for Query Relevance Scoring

The Cross-Encoder component is essential for evaluating the relevance of the generated questions. As indicated in the Figure 3.2, this module takes multiple inputs and produces relevance scores for each question.

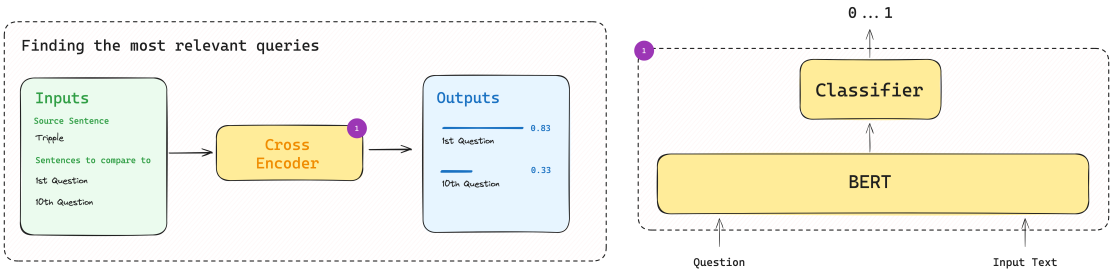


Figure 3.2: Cross-Encoder component, which assesses the relevance of generated questions by taking multiple input pairs and assigning a relevance score to each, supporting question evaluation and refinement.

Key features of the Cross-Encoder include:

- **Input Processing:**
  - Source Sentence: The original input text.
  - Sentences to Compare: Likely the 10 generated questions.

- **Scoring Mechanism:** The Cross-Encoder assigns numerical scores (*e.g.*, 0.83 as shown in the figure 3.2) to each question, indicating its relevance to the source sentence.
- **Comparative Analysis:** By processing all inputs simultaneously, the Cross-Encoder can perform nuanced comparisons between the original input and each generated question, as well as among the questions themselves.

In this case we use the *jinaai/jina-reranker-v1-turbo-en*[1] model from the Hugging Face Transformers library. This model is designed for blazing-fast reranking while maintaining competitive performance. It leverages the power of

[1] `https://huggingface.co/jinaai/jina-reranker-v1-turbo-en`

JinaBERT [9] model as its foundation. The model employs a process known as knowledge distillation to attain exceptional speed and efficiency, making it the optimal selection for our pipeline.

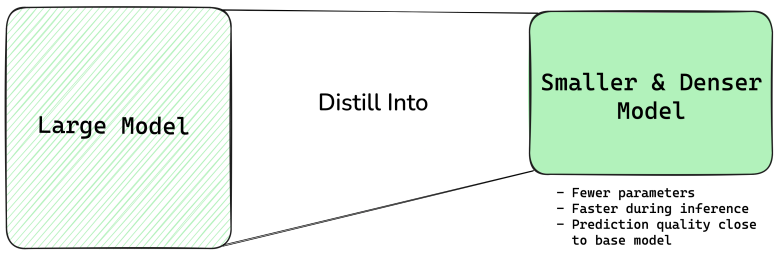


Figure 3.3: Knowledge Distillation Process for Enhanced Re-Ranking Efficiency

3.2.4 Relevance Threshold and Sorting

Following the Cross-Encoder's scoring, the pipeline implements a crucial decision point:

- **Sorting:** Questions are sorted based on their relevance scores, establishing a priority order for further processing.
- **Threshold Evaluation:** The system checks if the top question's score exceeds an upper threshold. This step ensures that only sufficiently relevant questions proceed further in the pipeline.
- **Feedback Loop:** If the threshold is not met, the process must loop back to generate new questions to adjust the existing ones, maintaining the quality of queries entering subsequent stages.

Table 3.2: Example of generated questions by the Gemma2 model with relevance scores assigned by the Jina Re-ranker Cross-Encoder.

Input	Question	score
Frédéric Passy award Nobel Peace Prize	Who was awarded the Nobel Peace Prize?	0.7458
	What award did Frédéric Passy receive?	0.7121
	Is Frédéric Passy a Nobel laureate?	0.8706
	In what category was the Nobel Peace Prize awarded to Frédéric Passy?	**0.9491**
	Who is known for receiving the Nobel Peace Prize?	0.5823
	What is the name of the award received by Frédéric Passy?	0.6318
	Is Frédéric Passy a recipient of the Nobel Prize in any field?	0.7652
	Who was recognized for his work towards peace?	0.1457
	What is the significance of the award given to Frédéric Passy?	0.6036
	Is there a Nobel laureate with the name Frédéric Passy?	0.8505

3.3 Information Retrieval Mechanisms

The Information Retrieval Mechanisms are an essential element of our pipeline, connecting query processing and content synthesis. This phase is tasked with gathering relevant data from internal and external sources, thereby establishing a comprehensive data repository for further analysis and response formulation. The mechanisms employed in this phase are designed to ensure breadth, depth, and relevance in the retrieved information.

3.3.1 Google Search Integration

A key feature of our information retrieval process is the integration of Google Search capabilities, as prominently displayed in the pipeline diagram. This integration serves to expand the information horizon beyond the confines of our internal Knowledge Graph Dataset. Key aspects of this integration include:

- **Query Submission:** The system submits the N top questions (where N is a predefined number) along with the main question to Google Search. This approach ensures a multi-faceted search that captures various aspects of the original query.
- **Result Fetching:** As indicated in the diagram, the system retrieves the top 100 search results. This number strikes a balance between comprehensiveness and computational efficiency.
- **Dynamic Information Access:** By leveraging Google Search, the system gains access to up-to-date information, complementing the more static nature of the internal Knowledge Graph.
- **Diverse Source Types:** Google Search results typically include a variety of source types (*e.g.*, websites, news articles, academic papers), enriching the diversity of the retrieved information.

Implementation considerations:

- The system may employ proxies or other mechanisms to manage rate limits and ensure uninterrupted access to search results.
- The search query may be customized based on the specific requirements of the pipeline, such as language restrictions, geolocation preferences, or result quantity. parameters are *lr*, *hl*, *gl*, and *num*.

Take note that the code base is generic, and it means that you can use any search engine, not only Google Search.

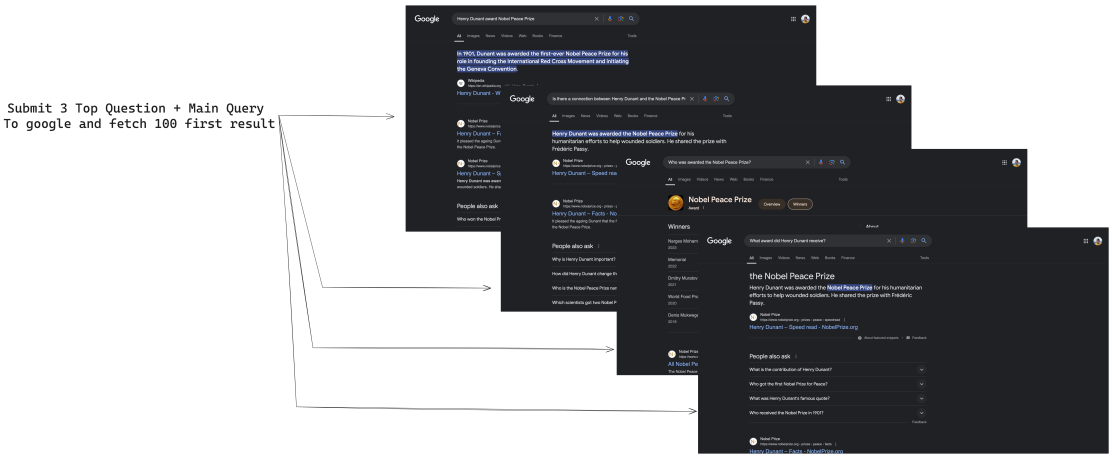


Figure 3.4: Fetching results from Google Search engine for top N questions and the main knowledge graph query.

3.3.2 Process and Extract Links

Following the retrieval of search results, the pipeline incorporates a crucial step of processing and extracting links from the gathered information. This process likely involves:

- **Parsing HyperText Markup Language (HTML) Content:** Extracting relevant textual information from the retrieved web pages.
- **Link Analysis:** Identifying and cataloging hyperlinks within the content, potentially uncovering additional relevant sources.

After Parsing the HTML content of the Google Search results, the system use HTML selectors to extract the links from the search results. In this case we also extract the title, url, description, price, date, duration, missing, rating, availability, and extra details from the search results. Some of the information mentioned above may not be available as it depends on the search results.

Then there is a need to crawl the extracted links to get the content of the page, for doing this we use the Python library called *GRequests*[2]. *GRequests* is a Python library that combines the power of gevent for asynchronous I/O with the simplicity of the Requests library for HTTP operations. It allows developers to perform concurrent HTTP requests easily, significantly speeding up operations that involve multiple API calls or web scraping tasks.

[2] `https://pypi.org/project/grequests/`

```python
import os
import grequests
from fake_useragent import UserAgent

ua = UserAgent(
    os=['windows'],
    browsers=["chrome", "edge", "firefox"],
    platforms=["pc"]
)

urls = [...List of Extracted URLs...]

rs = [
    grequests.get(u['url'],
    timeout=3, headers={"User-Agent": ua.random}) for u in urls
]
for index, response in grequests.imap_enumerated(rs, size=50):
    if response is None or response.status_code != 200:
        continue
    # Process the response content ...
```

Code 3.1: Crawling the Extracted URLs

There are several faults in the mentioned approach 3.1 that need to be addressed:

- **Site generated with javascript:** The provided approach does not handle sites that are generated with JavaScript and require dynamic rendering.
- **Protection against scraping:** Sites may have protection mechanisms against scraping, such as CAPTCHAs or IP blocking or behind spam protection services.
- **Login required:** Some sites require login credentials to access the content.

We can use the *Selenium*[3] library to handle the first issue, and for the second and third issues, we can use the *Scrapy*[4] library, but we stick with the provided approach for simplicity and speed.

With the extracted content, the system can now proceed to the next stage of the pipeline, where the information is further processed and analyzed. For extracting the content of the page, as our webpage are from vast sources, we need to use a robust and efficient library to extract the content of the page. We use

[3] https://www.selenium.dev/
[4] https://scrapy.org/

newspaper4k[5] library to extract the content of the page. *newspaper4k* is a Python library designed for extracting and parsing newspaper articles. It's an updated and improved version of the original newspaper3k library, offering enhanced functionality and compatibility with modern Python versions.

Key features of the *newspaper4k* library include:

- **Article Extraction:** Easily extract articles from news websites.
- **Multi-language Support:** Capable of processing articles in various languages.
- **Full-text Extraction:** Extracts the full text of articles, removing ads and extraneous content.
- **Keyword Extraction:** Automatically identifies key topics and keywords from articles.
- **Summary Generation:** Creates concise summaries of article content.
- **Metadata Parsing:** Extracts metadata such as authors, publication dates, and tags.
- **Image Extraction:** Identifies and extracts images associated with articles.

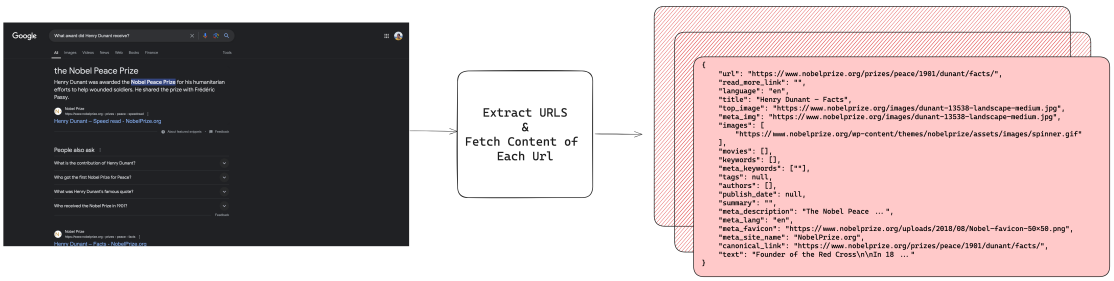


Figure 3.5: Extracted content from the crawled URLs using newspaper4k library.

For our purpose, we only use the text content of the page.

3.3.3 Data Pool Creation

The processed and extracted information culminates in the creation of a data pool, a centralized repository of relevant information that serves as the foundation for subsequent stages of the pipeline.

Key features of the data pool include:

[5] `https://newspaper4k.readthedocs.io/en/latest/`

- **Structured Storage:** Organizing the retrieved information in a format that facilitates efficient querying and analysis.
- **Source Diversity:** Maintaining a balance between information from web searches and the internal Knowledge Graph.
- **Relevance Scoring:** Potentially implementing a scoring system to prioritize more relevant or authoritative pieces of information within the pool.
- **Deduplication:** Employing mechanisms to identify and merge duplicate or highly similar pieces of information to reduce redundancy.

3.3.4 Data filtering

The data filtering process aims to refine our data pool by removing information from sources that are already part of creation of the Knowledge Graph. This ensures the uniqueness and independence of our data, for example, when working with the FactBench dataset, we exclude data from the following sources: wikipedia.org, wikimedia.org, wikidata.org and other similar wiki-based platforms.

On the other hand, we just keep the top 10 relevant information for the pipeline using cross-encoders discussed in Section 3.2.3, this way we ensure about the quality of the data.

Specifically:

- We start with a larger pool of data.
- We identify sources that are already the source of the Knowledge Graph.
- We remove any data points that come from these identified sources.

3.4 Embedding and Retrieval Tasks

An important step in our pipeline is the Embedding and Retrieval Tasks, which connect machine-interpretable vector representations to unprocessed textual input. This component is essential for improving the efficiency and accuracy of information retrieval and subsequent processing stages.

This section emphasizes embedding for retrieval tasks, specifically for small information segments, as depicted in the pipeline diagram.

3.4.1 Embedding Techniques for Smaller Chunks

The pipeline employs advanced embedding techniques to transform textual data into dense vector representations, facilitating more efficient and semantically aware retrieval processes.

Key aspects of this embedding process include:

- **Granularity:** The focus on "Smaller Chunks" suggests a fine-grained approach to embedding, where text is broken down into manageable units. This granularity allows for more precise retrieval and relevance assessment.
- **Dimensionality Reduction:** Transforming high-dimensional textual data into lower-dimensional vector spaces while preserving semantic relationships.

Challenges and Considerations

Several challenges and considerations are inherent in the Embedding and Retrieval Tasks:

- **Multilingual Support:** Extending embedding capabilities to support multiple languages, potentially through multilingual or language-agnostic models.
- **Embedding Interpretability:** Balancing the trade-off between the performance of dense embeddings and the interpretability often associated with sparse representations.
- **Temporal Dynamics:** Addressing the challenge of embedding temporally sensitive information and ensuring retrieval mechanisms account for time-based relevance.
- **Scalability:** Designing the embedding and retrieval systems to efficiently handle growing volumes of data without compromising on speed or accuracy.
- **Ethical Considerations:** Mitigating potential biases inherent in pre-trained embedding models and ensuring fair representation across diverse topics and perspectives.

3.4.2 Similarity Cutoff Strategy

One of the most important aspects that is highlighted in the pipeline diagram is the "Similarity Cutoff Strategy," which is significant since it is responsible for filtering and prioritizing the information that is included.

This strategy likely involves:

- **Threshold Definition:** Establishing a similarity threshold below which embedded chunks are considered insufficiently relevant or related to the query or context.
- **Similarity Metrics:** Employing appropriate similarity measures (*e.g.*, cosine similarity, Euclidean distance) to quantify the relatedness between embedded representations.
- **Re-run Mechanism:** Potentially re-do the response generation process if the similarity cutoff is not met for all the documents and the response is not generated.

The Similarity Cutoff Strategy serves several critical functions:

- **Noise Reduction:** Filtering out irrelevant or tangentially related information to improve the signal-to-noise ratio in subsequent processing stages.
- **Computational Optimization:** Reducing the volume of data processed in later stages, thereby enhancing overall system efficiency.
- **Relevance Enhancement:** Ensuring that only the most pertinent information is retained for query resolution and response generation.

It can be switched off if the dataset is small or if the similarity cutoff is not needed.

3.5 LLMs

LLMs play a pivotal role in our advanced NLP pipeline, serving as the cornerstone for sophisticated information synthesis and response generation. As illustrated in the pipeline diagram, multiple LLMs are employed, contributing to a robust and nuanced approach to question answering and information processing.

3.5.1 Integration of Multiple LLMs

The pipeline incorporates multiple LLMs working in concert, a design choice that offers several significant advantages:

- **Diversity of Perspectives:** By utilizing multiple models, the system can capture a broader range of interpretations and approaches to information synthesis.
- **Specialization:** Different LLMs may be fine-tuned or specialized for particular types of queries or domains, allowing for more targeted and accurate responses in specific contexts.
- **Robustness:** The multi-model approach provides redundancy and helps mitigate individual model biases or weaknesses.
- **Scalability:** Parallel processing of information through multiple LLMs can potentially improve the system's throughput and response time.

Implementation considerations:

- **Model Selection:** Choosing a diverse set of LLMs that complement each other in terms of strengths and specializations.
- **Load Balancing:** Implementing efficient mechanisms to distribute workload across the available models.
- **Version Management:** Maintaining and updating multiple LLMs to ensure they remain current and aligned with the latest advancements in NLP.

For running open-source LLMs, we use *Ollama*[6], Ollama is an open-source project that simplifies the process of setting up, running, and using LLMs locally on your machine. It provides a user-friendly area for managing and interacting with various LLMs, making it easier for developers and enthusiasts to experiment with AI without relying on cloud services. Under the hood, *Ollama* is just an API server written in go that serves GGUF models via llama.cpp [7] and a centralized hub of models/settings. Performance Considerations and Limitations of *Ollama*:

- Performance Considerations
  - Performance depends on your hardware, especially CPU and GPU capabilities.

[6] `https://ollama.com/`
[7] `https://github.com/ggerganov/llama.cpp`

  - Larger models require more RAM and storage space.
  - GPU acceleration can significantly improve inference speed.

- Limitations

  - Resource intensive for larger models.
  - May not match the performance of cloud-based solutions for some use cases.
  - Limited to models that are compatible with Ollama's framework.

3.5.2 Roles of LLMs in the Pipeline

Based on the pipeline diagram, the LLMs serve several crucial functions:

- **Information Synthesis:** Integrating and coherently combining information from various sources.
- **Context Processing:** Analyzing and interpreting the broader context of queries and retrieved information to generate more accurate and relevant responses.
- **Reasoning and Inference:** Drawing logical conclusions and making inferences based on the available information, potentially finding the correctness of the knowledge graph.

The prompt template used for RAG approach reported in Appendix A.3, we use the selected chunks from the previous steps to feed the LLMs context and also provide few examples to the LLMs to generate the better response.

3.6 Voting System and Conflict Resolution

Our state-of-the-art pipeline for natural language processing relies heavily on the integration of many models and the deployment of procedures to resolve conflicts. To ensure robust and trustworthy outcomes, this section covers the ways adopted to resolve disputes and harness model diversity.

3.6.1 Majority Voting System

A significant aspect of the LLM integration, is the establishment of a majority voting mechanism for response creation.

This method provides numerous advantages:

- **Consensus Building:** By aggregating outputs from multiple models, the system can identify areas of agreement, potentially leading to more reliable responses.
- **Error Mitigation:** Outlier responses or errors from individual models can be identified and potentially filtered out through the voting process.
- **Confidence Scoring:** The degree of consensus among models can serve as a proxy for the confidence level of the generated response.
- **Handling Ambiguity:** In cases where there's no clear majority, the system can potentially flag the response as uncertain or requiring further clarification.

Implementation challenges:

- **Weighting Mechanism:** Determining whether all LLMs should have equal weight in the voting process or if some models should be prioritized based on their specific strengths or reliability.
- **Threshold Setting:** Establishing the criteria for what constitutes a "majority" and how to handle cases with no clear consensus.
- **Combining Diverse Outputs:** Developing methods to meaningfully aggregate potentially disparate outputs from different models into a coherent final response.

The pipeline incorporates several mechanisms for resolving conflicts that may arise from divergent model outputs.

3.6.2 Conflict Resolution Strategies

As previously discussed 3.6.1, the system employs a majority voting approach among LLMs to determine the most appropriate response. This serves as a primary conflict resolution mechanism, leveraging the wisdom of the collective to mitigate individual model errors or biases.

Final Judge Implementation:

A crucial component in the conflict resolution process is the "final judge" module, as indicated in the pipeline diagram. This element plays a pivotal role in resolving conflicts and ensuring coherence in the system's outputs.

Key aspects of the final judge implementation:

- **Conflict Identification:** Detecting discrepancies or contradictions in outputs from different models.
- **Resolution Mechanisms:** Using predefined rules or algorithms to determine the final output based on the majority voting results.
- **Tie-Breaking:** Implementing strategies for scenarios where the majority voting system results in a tie or lacks a clear consensus.
- **Consistency Enforcement:** Ensuring that the final output maintains logical consistency and aligns with established knowledge bases.

The final judge module used a system named *Adaptive Conflict Resolution*, which is an adaptive approach to conflict resolution based on the specified settings.

Adaptive Conflict Resolution

The pipeline demonstrates an adaptive approach to conflict resolution, as evidenced by the following feature: "When two of the language models believe the answer is correct, and two believe it is wrong, we switch to a more advanced model, such as a commercial one or a model with more parameters." There are two ways to apply this adaptive method: 1) Directly selecting commercial models (*e.g.*, GPT-4), or 2) Using a more advanced, open-source model with more parameters (*e.g.*, Gemma2:27b). For the second option, we use the algorithm shown in Algorithm 1. This algorithm uses the mapping between the already used and final models to select the final model. It utilizes parameters to identify either the most consistent model or the least consistent model to choose based on a majority vote across the entire system.

This adaptive strategy offers several advantages:

- **Escalation Mechanism:** Provides a structured approach for handling ambiguous cases where simpler resolution methods are insufficient.
- **Resource Optimization:** Reserves the use of more advanced (and potentially more computationally expensive) models for cases that truly require their capabilities.

- **Accuracy Enhancement:** Leverages more sophisticated models to resolve complex conflicts, potentially leading to higher-quality outputs in challenging scenarios.
- **Flexibility:** Allows for the integration of specialized or proprietary models in a targeted manner, enhancing the system's overall capabilities without relying on these models for every query.

Implementation challenges:

- **Threshold Definition:** Determining the exact criteria for when to invoke the more advanced models.
- **Model Selection:** Choosing which advanced model to use based on the nature of the conflict and the query context.
- **Integration:** Ensuring smooth handover and result incorporation from the advanced models back into the main pipeline.

Algorithm 1 Resolve Ties in Majority Voting System

Require:
$modelScores$ - A list of consistency scores for each model
$finalJudger$ - A dictionary mapping file indices to final model
$atLeast$ - A boolean flag:
True: select model with the highest agreement score
False: select model with the lowest agreement score

```
procedure ProcessFiles(modelScores, finalJudger, atLeast)
    if atLeast is True then
        maxScore ← maximum score in modelScores
        candidates ← indices with score equal to maxScore
    else
        minScore ← minimum score in modelScores
        candidates ← indices with score equal to minScore
    end if
    chosenIndex ← random choice from candidates
    return finalJudger[chosenIndex]
end procedure
```

3.7 Performance Report

In Tables 3.3, 3.4, we provide a detailed performance report, highlighting key metrics that demonstrate the pipeline's efficiency. Other sections of the pipeline, such as RAG and conflict resolution mechanisms, will be evaluated in future chapters.

Table 3.3: Performance of our LLM-based tasks in production, generated by Openlit [8] with *Gemma2* model.

Task	Avg. Request Time*	Avg. tokens per request
Human understandable text 3.2.1	1.3164 sec	343.16
Question Generation 3.2.2	9.6076 sec	672.58

* The average time is calculated on the Macbook Pro with M2 max chip and 32GB of RAM.

Table 3.4: Performance of our information retrieval mechanisms.

Task	Avg. Time*
Sorting the questions by similarity 3.2.3	0.013 sec
Get documents (Google pages) 3.3.1	3.6 sec
Fetch documents for each triple 3.3.2	350 sec

* The average time is calculated on the unix based server with 2 cores and 4GB of RAM.

3.8 Pipeline Flow and Decision Points

The architecture of our pipeline is characterized by a sophisticated flow of information and a series of critical decision points. This structure enables the system to process complex queries, retrieve and synthesize relevant information, and generate accurate responses. This section provides a comprehensive analysis of the pipeline's flow and the key decision points that guide the processing of information.

The pipeline flow can be broadly categorized into several main stages, each with its own set of processes and decision points:

- Input Processing and Query Generation
- Information Retrieval and Enrichment
- Embedding and Relevance Assessment
- Multi-Model Processing and Synthesis
- Conflict Resolution and Final Output Generation

Let's examine each of these stages in detail in table 3.5, focusing on the flow of information and the critical decision points within each.

Several challenges and considerations are associated with managing the pipeline flow and decision points:

Table 3.5: Comprenhensive list of decision points in the pipeline flow.

Component	Decision Point
Input Processing and Query Generation	
Knowledge Graph Dataset Integration	Extent and nature of knowledge graph integration based on input complexity.
Human-Understandable Text Generation	Selection of the most appropriate natural language generation technique based on input and context.
Question Generation	Assessment of the quality and relevance of generated questions.
Cross-Encoder for Query Relevance	Determination of whether the top question score exceeds the upper threshold.
Information Retrieval and Enrichment	
Information Source Selection	Determining the most relevant and reliable sources for information retrieval.
Google Search Integration	Balancing between the breadth of search (number of questions submitted) and depth (number of results retrieved).
Process and Extract Links	Determining the relevance and quality of extracted information for inclusion in the data pool.
Data Pool Creation	Structuring the data pool for optimal accessibility in subsequent stages.
Embedding and Relevance Assessment	
Embedding for Retrieval Tasks	Selection of the most appropriate embedding technique based on the nature of the data.
Similarity Cutoff Strategy	Determination of the similarity cutoff threshold.
Context Processing	Determining the optimal chunk size and processing method.
Multi-Model Processing and Synthesis	
Parallel LLM Processing	Allocation of specific tasks or aspects of the query to different models based on their strengths.
Synthesis of Information	Determination of the method of synthesis (e.g., concatenation, abstraction, or hybrid approaches).
Majority Voting System	Assessment of the level of agreement among models.
Conflict Resolution and Final Output Generation	
Conflict Identification	Determination of the threshold for what constitutes a significant conflict requiring resolution.
Adaptive Model Selection	When two models believe the answer is correct and two believe it's wrong, switching to a more advanced model.
Selection of Commercial Models	Choose the commercial model based on the user specified settings.
Final Judge Implementation	Determination of the final response based on aggregated model outputs and conflict resolution results.
Response Generation	Selection of the most appropriate format and level of detail for the response.

- **Computational Efficiency:** Balancing the depth of processing at each stage with the need for timely responses.
- **Error Propagation:** Ensuring that errors or biases introduced at early stages don't disproportionately affect the final output.
- **Adaptability:** Designing decision points that can adapt to different query types and complexity levels.
- **Transparency:** Maintaining traceability of decisions made throughout the

pipeline for accountability and debugging purposes.

- **Scalability:** Ensuring that the pipeline can handle increasing query volumes without significant degradation in performance or accuracy.

In conclusion, the Pipeline Flow and Decision Points represent a complex yet well-structured approach to natural language processing and question answering. By implementing a series of carefully designed stages and critical decision points, the system aims to process information in a manner that maximizes accuracy, relevance, and reliability. The adaptive nature of the pipeline, particularly in its approach to conflict resolution and model selection, demonstrates a commitment to handling a wide range of query complexities and scenarios. However, the intricate nature of this flow also underscores the importance of ongoing optimization, monitoring, and refinement to ensure that the system continues to perform effectively and ethically in the face of evolving challenges and requirements in the field of AI and natural language processing.

3.9 Ethical Considerations and Limitations

The development and deployment of advanced pipelines, such as the one described in this thesis, necessitate a thorough examination of ethical considerations and an acknowledgment of system limitations. This section explores the ethical implications of our fact-checking system and discusses its inherent constraints.

3.9.1 Ethical Considerations

While the pipeline diagram does not explicitly highlight ethical components, several aspects of the system raise important ethical considerations:

Misinformation and Harmful Content

The system's ability to synthesize information from various sources poses risks related to misinformation:

- **Propagation of False Information:** Potential for the system to inadvertently spread misinformation present in retrieved data.
- **Generation of Harmful Content:** Risk of producing responses that could be considered harmful, offensive, or inappropriate.

Environmental Considerations

The computational resources required to run multiple LLMs and process large volumes of data raise environmental concerns:

- **Energy Consumption:** High energy usage associated with running complex AI models and large-scale data processing.
- **Carbon Footprint:** Environmental impact of the infrastructure required to support the pipeline.

3.9.2 Limitations

Understanding and acknowledging the limitations of the system is crucial for ethical deployment and user trust:

Table 3.6: Limitations of the pipeline, categorized by scope of knowledge.

Limitation	Description
	Scope of Knowledge
Temporal Limitations	The system's knowledge base and models have a cutoff date, potentially leading to outdated information.
Domain Specificity	Gaps in specialized or niche areas of knowledge may limit performance.
	Language and Cultural Limitations
Language Coverage	Biases towards languages well-represented in training data, challenges in handling less common languages.
Cultural Context	Limitations in understanding and responding to culturally specific queries or contexts.
	Reasoning and Inference Capabilities
Complex Reasoning	Challenges in handling queries requiring advanced logical reasoning or domain-specific expertise.
Causal Understanding	Difficulties in inferring causal relationships.
	Handling of Ambiguity and Context
Contextual Nuances	Difficulties in capturing subtle contextual cues that humans naturally understand.
Disambiguation	Challenges in resolving ambiguities in queries without additional user input.
	Real-time Adaptation
Static Knowledge Base	Limitations in adapting to real-time changes without system updates.
Learning from Interactions	Inability to learn and improve from individual user interactions due to privacy and architectural constraints.

4

Empirical Evaluation

4.1 Dataset Analysis

This section presents an analysis of the three datasets used in our empirical evaluation: *FactBench*, *YAGO*, and *DBpedia*. Each dataset offers unique characteristics and challenges, providing a comprehensive basis for assessing our knowledge graph fact verification system.

4.1.1 FactBench Dataset

FactBench is a multilingual dataset specifically designed for fact-checking in knowledge graphs [1] [8]. It comprises 2,800 facts, 1.500 true and 1.300 false, across three languages: English, German, and French. The dataset covers various domains, including geography, politics, and entertainment. The data was automatically extracted from Wikipedia[2] (DBpedia respectively) and Freebase[3].

To obtain positive examples, the authors leverages facts from both DBpedia and Freebase. For each property under consideration, they generated these examples by issuing either a SPARQL (for DBpedia) or MQL (for Freebase) query. They then selected the top 150 results. In Freebase, results are ranked using an internal relevance score, while in *DBpedia*, the results are sorted by

[1] https://github.com/DeFacto/FactBench
[2] https://www.wikipedia.org/
[3] https://developers.google.com/freebase

the number of inbound links to the resource's corresponding Wikipedia page. In total, 1500 correct statements were collected, with 750 allocated to both the test and training sets, ensuring that each relation had 150 positive facts equally distributed between the test and training sets.

For generating incorrect facts (negative examples), the authors modified correct ones while adhering to domain and range constraints. To ensure that the negative examples closely resemble true statements (*i.e.*, meaningful triples), the team altered the positive examples while still adhering to domain and range restrictions. Given a triple (s, p, o) and its timespan (from, to) from the knowledge base, they used different methods to generate sets of negative examples. These methods include modifying the subject, object, both subject and object, or the property. Additionally, they included random modifications, a 20% mix of these methods, and variations in the date.

We don't consider the time aspect in our evaluation, as our system is not designed to handle time-sensitive issues. We consider a configuration where incorrect facts are a mix produced using different negative example generation strategies, resulting in a ground-truth accuracy of $\mu = 0.54$. Key characteristics of *FactBench* include:

- Multilingual support (English, German, and French)
- Diverse fact types, including domain-specific and temporal facts
- Manually curated for high-quality ground truth

In our analysis, we found that *FactBench* presents a balanced challenge for our system, with a mix of straightforward and complex fact verification tasks.

4.1.2 YAGO Dataset

YAGO (Yet Another Great Ontology) is a large-scale knowledge base derived from Wikipedia, WordNet [4], and GeoNames [5]. For our evaluation, we use *YAGO2-sample* [6] [23], a subset of the full YAGO2 knowledge graph derived from AMIE horn clauses [7]. This sample consists of 1,386 beliefs spanning 16 unique predicates.

Key characteristics of the *YAGO* dataset in our evaluation include:

[4] `https://wordnet.princeton.edu/`
[5] `https://www.geonames.org/`
[6] `https://aclanthology.org/attachments/D17-1183.Attachment.zip`

- High accuracy: The gold standard accuracy of the *YAGO2-sample* is 99.20%, indicating a very high-quality dataset.
- Diverse predicates: The sample covers 16 different predicates, allowing for evaluation across a range of relationship types.
- Balanced distribution: Unlike domain-specific datasets, *YAGO2-sample* covers a broad range of topics, reflecting the diverse nature of Wikipedia.

The high accuracy of the *YAGO2-sample* presents a unique challenge for our evaluation system.

4.1.3 DBpedia Dataset

DBpedia serves as a comprehensive, large-scale knowledge base derived from Wikipedia, offering structured information about millions of entities. For our evaluation, we utilize *DBpedia* version 2015-10 which contains approximately 6.2M entities and 1.1B triplets. Following Marchesin et al.'s approach [19] to entity-oriented research, several filtering criteria were applied to ensure high-quality data for evaluation.

The analysis was restricted to subject entities that include both: 1) rdfs:label predicate and 2) rdfs:comment predicate

Additionally, they focused exclusively on A-Box triplets (assertional knowledge) while excluding T-Box triplets (terminological knowledge). The T-Box encompasses ontological entities and relationships, while A-Box contains the actual assertions that need verification. After applying these filters, their working dataset consisted 4.6M entities with 170M triplets.

From this filtered dataset, they conducted a comprehensive annotation study on 9,930 facts, which were carefully selected to represent diverse types of relationships and knowledge domains within *DBpedia*. To ensure annotation quality, Marchesin et al. implemented several measures:

- Multiple annotators per fact (minimum of three annotations per triplet)
- Expert consensus requirement for final labels
- Binary validation approach treating all incorrect facts equally regardless of error type
- Documented agreement rates between expert annotators (77% agreement with Cohen's score of 0.51)
- Third-party resolution for 82% of initial disagreements

The dataset was carefully curated to ensure a manageable yet representative evaluation set, derived from the vast scale of *DBpedia*. By utilizing this curated subset of *DBpedia*, we benefit from a balance between the richness of a real-world knowledge graph and the practicality required for thorough empirical evaluation. For our evaluation system, we use subset of this annotated dataset, by removing facts with <*UNK*> labels, resulting in 9,344 facts.

Dataset Summary

To conclude, our empirical evaluation utilizes three distinct datasets: *FactBench*, *YAGO*, and *DBpedia*. Each dataset offers unique characteristics that allow us to assess our knowledge graph veracity framework across diverse scenarios. Table 4.1 summarizes the key features of these datasets:

Table 4.1: Statistical summary of FactBench, YAGO, and DBpedia datasets

	FactBench	**YAGO**	**DBpedia**
Num. of Facts	2,800	1,386	9,344
Num. of Predicates	10	16	1,092
Avg. Facts per Entity	2.42	1.69	3.18
Gold Accuracy (μ)	0.54	0.99	0.85

FactBench provides a good distribution of true and false statements across multiple domains, offering a robust testbed for fact verification. *YAGO*, with its high accuracy, challenges our framework to detect subtle inaccuracies in an otherwise highly reliable knowledge graph. The *DBpedia* subset, curated specifically for entity-oriented search tasks, allows us to evaluate our framework in the context of query-dependent fact checking. This diverse selection of datasets enables a comprehensive evaluation of our veracity estimation framework.

4.2 Candidate Models

4.2.1 Gemma2

Gemma is a family of lightweight, state-of-the-art open models from Google, built from the same research and technology used to create the Gemini models [7].

[7] https://deepmind.google/technologies/gemini/#introduction

They are text-to-text, decoder-only large language models, available in English, with open weights for both pre-trained variants and instruction-tuned variants. Gemma 2 implements a similar architecture to the original Gemma model, with a few key differences. The model alternates between local sliding window attention with a 4096-token span and global attention with an 8192-token span in alternate layers. Logits are capped within a specified range to stabilize the values during attention and final layers, with soft_cap set to 50 for self-attention layers and 30 for the final layer. RMSNorm is used for normalization in transformer sub-layers, and Grouped-Query Attention (GQA) with two groups enhances inference speed without sacrificing performance. This hybrid approach aims to balance efficiency with the ability to capture long-range dependencies in the input.

We selected the *Gemma2-9B* model for our evaluation, which has 9 billion parameters. The 9B model learns from a larger teacher model during initial training in pre-training and use on-policy distillation to refine its performance post-training. This approach allows *Gemma2-9B* to capture the knowledge and capabilities of the larger model while maintaining a more compact size. As a result, *Gemma2-9B* delivers competitive performance relative to models 2-3 times its size, making it an attractive choice for applications with computational constraints.

4.2.2 Qwen2.5

Qwen2.5 is the latest series of Qwen LLMs [37]. For *Qwen2.5*, Alibaba Cloud [8] release a number of base language models and instruction-tuned language models ranging from 0.5 to 72 billion parameters. All models are pre-trained on our latest large-scale dataset, encompassing up to 18 trillion tokens. Compared to *Qwen2*, *Qwen2.5* has acquired significantly more knowledge and has greatly improved capabilities in coding and mathematics. Additionally, the new models achieve significant improvements in instruction following, generating long texts (over 8K tokens), understanding structured data (*e.g.*, tables), and generating structured outputs especially JSON. *Qwen2.5* models are generally more resilient to the diversity of system prompts, enhancing role-play implementation and condition-setting for chatbots. Like *Qwen2*, the *Qwen2.5* language models

[8] `https://www.alibabacloud.com/en?_p_lc=7`

support up to 128K tokens and can generate up to 8K tokens. They also maintain multilingual support for over 29 languages [32].

We selected the *Qwen2.5-7b* model for our evaluation, which has 7 billion parameters.

4.2.3 Llama3.1

The Meta *Llama3.1* collection of multilingual LLMs is a collection of pre-trained and instruction tuned generative models in 8B, 70B and 405B sizes (text in/text out). The *Llama3.1* instruction tuned text only models (8B, 70B, 405B) are optimized for multilingual dialogue use cases and outperform many of the available open source and closed chat models on common industry benchmarks.

Llama3.1 is an auto-regressive language model that uses an optimized transformer architecture. *Llama3.1* was pre-trained on 15 trillion tokens of data. In post-training The models produced by doing several rounds of alignment on top of the pre-trained model. Each round involves Supervised Fine-Tuning (SFT), Rejection Sampling (RS), and Direct Preference Optimization (DPO). Meta use synthetic data generation to produce the vast majority of our SFT examples, iterating multiple times to produce higher and higher quality synthetic data across all capabilities. Additionally, they invest in multiple data processing techniques to filter this synthetic data to the highest quality. This enables model to scale the amount of fine-tuning data across capabilities. Compared to previous versions of Llama, developers improved both the quantity and quality of the data we use for pre and post-training. These improvements include the development of more careful pre-processing and curation pipelines for pre-training data, the development of more rigorous quality assurance, and filtering approaches for post-training data [6, 1].

We selected the *Llama3.1-8b* model for our evaluation, which has 8 billion parameters.

4.2.4 Mistral

The *Mistral* model, released by Mistral AI [9], is a high-performance LLM, designed to outperform larger models in efficiency and effectiveness. With

[9] `https://mistral.ai/`

innovations such as GQA and Sliding Window Attention (SWA), Mistral offers faster inference and better handling of long sequences, reducing computation costs while maintaining high performance [12, 20].

We selected the *Mistral-7b* model for our evaluation, which has 7.3 billion parameters, its structure allows it to be both cost-effective and memory efficient, making it suitable for a wide variety of real-world applications

Candidate Model Summary

The selection of candidate models for our system was guided by the need for diversity, efficiency, and reliability in processing fact verification tasks within knowledge graphs. We chose *Gemma2*, *Qwen2.5*, *Llama3.1*, and *Mistral* for their specific strengths in handling diverse linguistic queries, reasoning capabilities, and compatibility with RAG pipelines. Each of these models brings unique advantages to our verification framework, as summarized in Table 4.2.

Table 4.2: Summary of key strengths of selected candidate LLMs for knowledge graph fact verification.

Model	Key Strengths	Description
Gemma2	Dense Retrieval & Query Processing	Optimized for dense retrieval tasks, Gemma2 processes complex linguistic structures, making it well-suited for entity-rich query generation and document ranking.
Qwen2.5	Logical Reasoning & Prompt Efficiency	Excels in reasoning tasks with minimal prompting. Its accuracy in logical inference supports consistent veracity assessments, especially for ambiguous or conflicting evidence.
Llama3.1	Efficiency & Versatility	Offers a balance of efficiency and accuracy, with robust performance across fact-checking benchmarks. Llama3.1's lower computational demands ensure responsive processing without compromising output quality.
Mistral	Context Sensitivity & Interpretability	Known for nuanced, context-driven outputs and interpretability. Mistral's language generation capabilities provide clear, human-like explanations, making it ideal for understanding the facts like a human.

For model selection, we can choose either instruction-tuned and quantized models with similar architectures or with different architectures. Here, we opted for models with varied architectures to make the ensemble more versatile and capable of handling a wide range of query scenarios. Using diverse models

in the ensemble offers a balanced approach to complex fact verification tasks across knowledge graphs. This multi-model setup enhances adaptability and reliability, allowing the system to respond accurately to diverse verification scenarios, even when they differ in nature.

4.3 Experimental Setup

4.3.1 Performance Metrics and Evaluation

Performance metrics are essential in assessing the efficacy, efficiency, and reliability of a system or model. The selection of metrics mostly depends on the characteristics of the task, the data, and the objectives. This section emphasizes the principal performance metrics typically employed in systems utilizing LLMs, information retrieval, and various machine learning tasks.

Correct and Incorrect Criteria

The system incorporates explicit CORRECT and INCORRECT states, indicating a binary evaluation mechanism for overall performance. This fundamental assessment provides a clear, high-level indication of the system's success in handling queries.

Relevance and Accuracy Metrics

The evaluation of a fact-checking system typically involves assessing both the correctness and relevance of responses.

Potential metrics include:

- **F1 Score:** The harmonic mean of precision and recall, providing a balanced measure of accuracy.
- **Accuracy:** The proportion of correct responses generated by the system.

Latency and Efficiency Measures

Given the complexity of the pipeline, evaluating its operational efficiency is crucial:

- **Response Time:** Measuring the end-to-end time from query input to response generation.
- **Component-wise Latency:** Assessing the processing time of individual pipeline components (*e.g.*, embedding generation, LLM processing). Fully reported in ablation study in chapter 5 and performance report in section 3.7.
- **Cost Efficiency:** Evaluating the cost-effectiveness of the pipeline in terms of computational resources and infrastructure by reporting the average token used per query.

Consistency Evaluation

The use of multiple models and a conflict resolution mechanism necessitates specific evaluation of output consistency:

- **Stability Across Models:** Assessing the consistency of responses generated by different LLMs for the same query, refer to Algorithm 2.

Algorithm 2 Calculate Model Consistency Per Model

procedure ModelStabilityCal($models$) ▷ Containing binary results
  $m\_len \leftarrow$ length($models$), $stabilityScores \leftarrow []$
  for $i \leftarrow 0$ to $m\_len - 1$ **do**
    $m1 \leftarrow models[i]$, $mStabilities \leftarrow []$
    for $j \leftarrow 0$ to $m\_len - 1$ **do**
      if $i \neq j$ **then**
        $m2 \leftarrow models[j]$, $matchCount \leftarrow 0$
        $totPreds \leftarrow$ length($m1$)
        for $k \leftarrow 0$ to $totalPredictions - 1$ **do**
          if $m1[k] = m2[k]$ **then**
            $matchCount \leftarrow matchCount + 1$
          end if
        end for
        $mStabilities \leftarrow \frac{matchCount}{totPreds}$ ▷ Append the stability score
      end if
    end for
    $stabilityScores[i] \leftarrow$ mean($mStabilities$)
  end for
  return $stabilityScores$ ▷ Dictionary with model stability scores
end procedure

4.3.2 System Configurations

The system configurations are selected based on the best results obtained from black-box testing the pipeline through a series of experiments, detailed in chapter 5. Table 4.3 summarizes the key system configurations used in our empirical evaluation.

Table 4.3: System configurations for empirical evaluation

Section	Parameter	Considerations
Human Understanble Text	Gemma2:9b	Other LLMs can be used, but using instruction-tuned models is recommended. This is skipped for *FactBench* dataset as discussed on 3.2.1.
Question Generation	Gemma2:9b	Other LLMs can be used, but using instruction-tuned models is recommended.
Question Relevance	Jina-reranker-v1-turbo-en	Cross-encoder models are recommended for this task.
Question RelevanceThreshold	0.5	–
Num. of Selected Questions	3	–
Google Search	–	Used query params: *lr* = 'lang_en', *gl* = 'us', *hl* = 'en', *num* = '100'. The lr parameter is set to the language of the query, gl to the country, hl to the language, and num to the number of results.
Num. of Selected Documents	10	–
Document Selection	ms-marco-MiniLM-L-6-v2	Filtered out the documents from these origins: dbpedia, wikipedia, wikimedia, wikidata, quora, britannica, scholarpedia, newworldencyclopedia, everipedia, encyclopedia, wikibooks, wiktionary, wikiversity, wikisource, wikiquote, wikivoyage, academia, and nytimes
Embedding Model	bge-small-en-v1.5	–
Chunking Strategy	Sliding Window window size 3	–
Similarity Cut-off	Simple	Use the threshold to filter out irrelevant documents.
Similarity Cut-off Threshold	0.3	–
Top_k	6	–

Tie-Breaking	–	Use model with higher-param for each model, for llama3.1:8b → 70b, gemma2:9b → 27b, qwen2.5:7b → 14b, and mistral:7b → mistral nemo:12b.

The tests are run on a server with the following specifications:

- **Model Name:** Mac Studio
- **Model Identifier:** Mac14,14
- **Model Number:** Z180000M3T/A
- **Chip:** Apple M2 Ultra
- **Total Number of Cores:** 24 (16 performance and 8 efficiency)
- **Memory:** 192 GB
- **System Firmware Version:** 11881.1.1
- **OS Loader Version:** 11881.1.1

4.4 Comparative Analysis

4.4.1 Discussion of Results

Evaluation across three distinct datasets - *FactBench*, *YAGO*, and *DBpedia* - shows significant insights into the performance and efficiency of different language models for knowledge graph fact verification.

Based on Table 4.4, in the *FactBench* dataset, *Gemma2* emerged as the strongest individual performer, achieving an accuracy of 0.9014 and an F1 score of 0.9085. These results were further enhanced by the ensemble approach using *Qwen2.5:14b*, which improved the accuracy to 0.9057 and F1 score to 0.9145. On the *YAGO* dataset, Mistral demonstrated exceptional performance, reaching an accuracy of 0.9221 and an F1 score of 0.9594. The consistent high performance across models on this dataset suggests that well-structured, high-quality knowledge graphs can be effectively verified using our approach. The ensemble method

Table 4.4: Empirical evaluation results of the proposed system and candidate LLMs over the FactBench, YAGO, and DBpedia.

ms-marco-MiniLM-L-6-v2, BAAI/bge-small-en-v1.5, Sliding Window (ws 3), Similarity Cut-off (Original), Top_k 6

Dataset	Model	Consistency*	Avg. request time	Avg. input tokens	Avg. output tokens	Acc	F1
FactBench	Gemma2	0.8738	2.32s	1509.30	19.95	0.9014	0.9085
	Qwen2.5	0.8747	2.46s	1509.18	67.73	0.8746	0.8910
	LLama3.1	0.8296	2.87s	1509.24	104.65	0.8243	0.8378
	Mistral	0.8686	1.73s	1509.17	8.81	0.8507	0.8729
	Most (Qwen2.5:14b)	0.9194	27.66s	1509.18	57.44	0.9057	0.9145
	Least (Llama3.1:70b)	0.9195	12.70s	1749.14	8.73	0.9025	0.9124
YAGO	Gemma2	0.8882	2.15s	1508.83	17.06	0.8506	0.9191
	Qwen2.5	0.8884	2.50s	1509.35	72.87	0.8600	0.9246
	LLama3.1	0.8552	7.20s	1509.25	104.58	0.8333	0.9089
	Mistral	0.8920	1.67s	1509.31	8.03	0.9221	0.9594
	Most (Mistral-nemo:12b)	0.9311	2.31s	1560.97	9.13	0.8701	0.9304
	Least (Llama3.1:70b)	0.9311	11.60s	1560.97	10.19	0.8853	0.9391
DBpedia	Gemma2	0.8207	7.80s	1551.69	27.15	0.6821	0.7865
	Qwen2.5	0.8247	2.63s	1552.02	73.41	0.7236	0.8211
	LLama3.1	0.7573	3.06s	1551.98	110.82	0.6224	0.7377
	Mistral	0.8162	1.81s	1551.95	8.76	0.7201	0.8192
	Most (Qwen2.5:14b)	0.8858	6.48s	1695.92	84.47	0.7014	0.8020
	Least (Llama3.1:70b)	0.8860	12.82s	1701.77	8.99	0.7099	0.8089

* Consistency score for each individual model is calculated across all models, excluding the ensemble models.

with *Mistral-nemo:12b* maintained strong results while providing additional verification confidence in ambiguous cases. The *DBpedia* dataset proved more challenging, with overall lower performance across all models. *Qwen2.5* achieved the best individual results with an accuracy of 0.7236 and an F1 score of 0.8211. This performance difference highlights the impact of dataset complexity and structure on verification accuracy.

Table 4.5: Statistical analysis of output tokens and request times per query across FactBench, YAGO, and DBpedia datasets for each used model.

	Gemma2		Qwen2.5		Llama3.1		Mistral		Mistral-nemo		Qwen2.5:14b		Llama3.1:70b	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
Output tokens	24.98	24.91	72.32	29.78	108.11	87.26	9.85	21.12	9.13	12.83	73.64	60.80	10.06	13.94
Request time	6.11s	51.38	2.59s	0.68	3.44s	19.59	1.79s	0.55	2.31	0.77	11.18s	68.12	13.94s	6.48

In terms of computational efficiency as showed in Tables 4.4, and 4.5 , Mistral demonstrated remarkable performance, generating minimal output tokens (average 8.81-9.85) while maintaining competitive accuracy. This efficiency is especially remarkable when compared to *LLama3.1*, which generated considerably more tokens (average 104.58-110.82) without any significant improvement in accuracy. Additionally, it suggests that *LLama3.1* often failed to follow instructions closely, opting to reason through every fact rather than verifying correctness. Input token counts remained relatively consistent across models, ranging from

approximately 1500 to 1700 tokens.

The processing time analysis reveals that *Mistral* consistently achieved the fastest request times (1.73-1.81s), while ensemble methods required longer processing times. However, it's important to note that ensemble methods were only employed for tie-breaking scenarios, affecting approximately 5–10% of the total queries. This selective application of ensemble methods effectively balances the trade-off between computational cost and accuracy improvement.

Table 4.6: Statistical analysis of request time per query across FactBench, YAGO, and DBpedia datasets.

	FactBench		**YAGO**		**DBpedia**	
	Avg	**Std**	**Avg**	**Std**	**Avg**	**Std**
Request time*	4.59s	20.17	4.55s	33.03	6.32s	39.04

* Time taken for the embedding phase and LLM request.

The analysis from Tables 4.4 and 4.6 show notable patterns in the performance specific to each dataset. Models generally achieved better results on more structured datasets like *FactBench* and *YAGO* compared to *DBpedia*. This pattern suggests that the clarity and consistency of the underlying knowledge graph significantly influence verification accuracy. Request times also varied across datasets, with *DBpedia* queries requiring longer processing times (6.32s) compared to *FactBench* (4.59s) and *YAGO* (4.55s), likely due to its greater complexity and size.

The ensemble approach proved particularly effective in resolving ambiguous cases. While the computational cost of ensemble methods is higher, their selective application only to uncertain cases (5-10% of queries) makes this trade-off acceptable in practice. The high consistency scores observed in ensemble methods (>0.91) suggest more reliable predictions for challenging cases, justifying the additional computational investment for these specific instances.

Since *DBpedia* proved to be the most challenging dataset in our evaluation, we performed a deeper analysis based on the dataset stratification established by Marchesin et al. [19]. Each knowledge graph triple was assigned to one of seven partitions, numbered 1–7, where partition 1 represents the least popular/common knowledge and partition 7 represents the most popular/common knowledge. This stratification helps us understand how our verification system performs across different levels of fact popularity and complexity within *DBpedia*. The analysis can reveal whether the system's accuracy varies between common, well-documented facts versus more obscure or specialized knowledge.

This insight is valuable for identifying areas where the system needs improvement and understanding its real-world applicability across different types of knowledge.

Table 4.7: Partition-wise evaluation results of the proposed system and candidate LLMs over the DBpedia dataset.

	Weight	Size	Gemma2		Qwen2.5		Llama3.1		Mistral		Qwen2.5:14b		Llama3.1:70b	
			Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Stratum 1	0.9120	2223	0.672	0.773	0.708	0.804	0.600	0.712	0.710	0.806	0.688	0.786	0.700	0.797
Stratum 2	0.0616	1695	0.697	0.796	0.720	0.816	0.627	0.738	0.726	0.820	0.711	0.807	0.720	0.814
Stratum 3	0.0177	1588	0.689	0.791	0.736	0.828	0.632	0.743	0.719	0.819	0.708	0.806	0.719	0.815
Stratum 4	0.0044	1327	0.689	0.797	0.737	0.835	0.628	0.747	0.709	0.815	0.706	0.811	0.714	0.817
Stratum 5	0.0029	1058	0.666	0.780	0.705	0.813	0.638	0.758	0.724	0.826	0.690	0.800	0.695	0.804
Stratum 6	0.0010	814	0.692	0.800	0.719	0.822	0.614	0.739	0.733	0.833	0.709	0.813	0.708	0.812
Stratum 7	0.0001	629	0.671	0.776	0.779	0.864	0.650	0.762	0.754	0.848	0.731	0.826	0.747	0.838

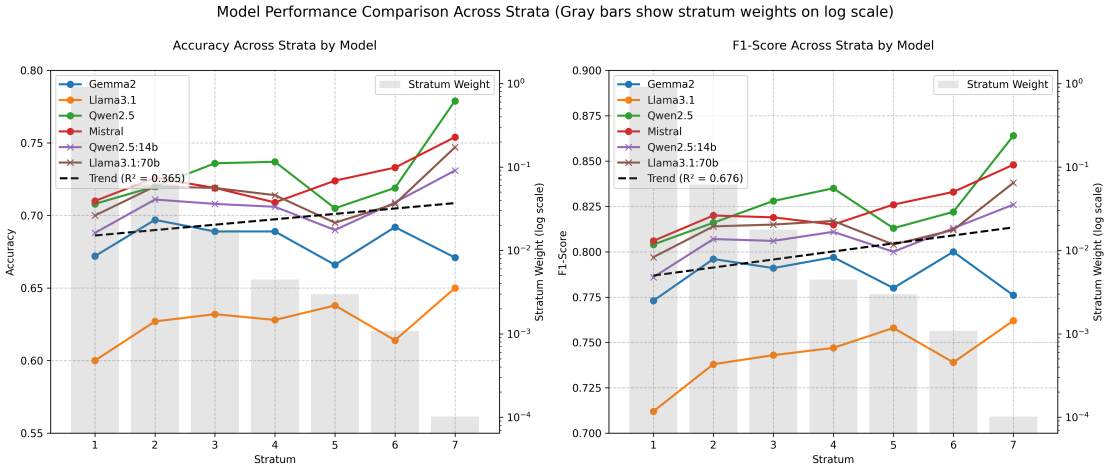


Figure 4.1: Partition-wise Model Performance Comparison: Accuracy and F1-scores for knowledge graph fact verification on DBpedia dataset. Gray bars indicate stratum weights (log scale).

Based on Table 4.7 and Figure 4.1, the models demonstrate a clear trend of improved performance on more common knowledge. *Qwen2.5* exhibits particularly strong performance, with accuracy increasing from 0.708 in Stratum 1 to 0.779 in Stratum 7, and F1-scores following a similar upward trajectory. This suggests that the model benefits from the richer context and more consistent representation of popular facts in the knowledge base.

The performance disparity between lower and higher strata highlights a common challenge in knowledge graph verification: the system's reliability varies with fact popularity. This insight is particularly valuable for real-world applications, where handling both common and specialized knowledge is crucial.

4.4.2 Qualitative Error Analysis

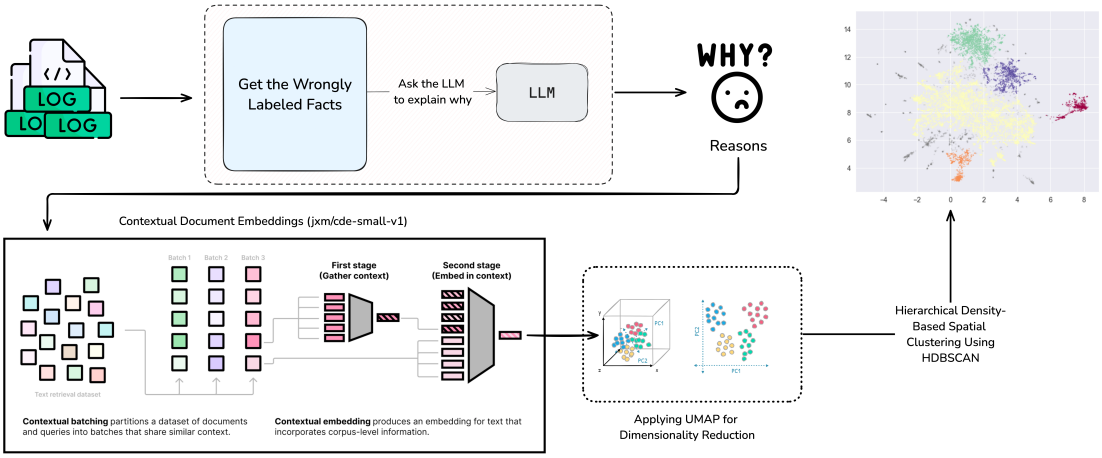


Figure 4.2: Collecting logs and leveraging LLM-generated reasoning, combined with contextual document embeddings (jxm/cde-small-v1), to cluster errors using a hierarchical density-based spatial technique.

As depicted in Figure 4.2, our objective is to categorize errors by LLMs and text embedding model. This approach helps reveal common error types and patterns by clustering explanations into distinct groups. The process begins by gathering logs of incorrectly labeled data, referred to here as "wrongly labeled facts." These logs are analyzed, and we then prompt an LLM to generate explanations, or "reasons," for each error. This step provides context and may highlight underlying patterns or causes that contribute to these errors. The prompt template used for this reasoning process is detailed in Appendix A.4. After obtaining explanations from the LLM, we use a specialized text embedding model named "jxm/cde-small-v1" [10]. We selected cde-small-v1 because it is the highest-ranked small model (under 400 million parameters) on the MTEB leaderboard for text embedding models, as of October 1, 2024. This model transforms each explanation into a contextualized embedding, capturing both semantic meaning and specific instruction-driven nuances for each error's context [21]. Next, these embeddings undergo dimensionality reduction using Uniform Manifold Approximation and Projection (UMAP), a technique that projects high-dimensional data into two or three dimensions while preserving local and some global structure. UMAP's visualization helps identify potential

[10] `https://huggingface.co/jxm/cde-small-v1`

clusters or groupings of similar errors, making it easier to observe patterns that might be difficult to see in higher dimensions. Once reduced in dimensionality, the embeddings are fed into Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN), a clustering algorithm well-suited for discovering clusters in data with varying density. HDBSCAN clusters the error embeddings based on their density, identifying groups of similar errors and isolating outliers. Following clustering, we identify some reasons from each dataset. These representative reasons are then provided to an LLM to assign descriptive labels to each error category, which encapsulate the main types of errors across the dataset.

The labeling data for each cluster is as follows:

- **UnLabeled:** The information or context provided does not contain the claimed details, such as references to specific individuals, places, or events that are purportedly associated with the topic.
- **Relationship Errors:** Errors arise from misstatements regarding relationships between people, such as marital status or religious affiliations that conflict with the provided details.
- **Role Attribution Errors:** Errors are due to incorrect associations of individuals with particular roles, places, or teams that do not match the details in the context.
- **Geographic/Nationality Errors:** This category includes errors related to locations, national affiliations, or settings that do not align with the context or provide contradictory information.
- **Genre/Classification Errors:** Misclassifications of films, genres, or roles are highlighted here, especially when certain works are wrongly associated with people, studios, or genres.
- **Identifier/Biographical Errors:** These errors involve incorrect identifiers or biographical details, such as award titles, label names, or authorship that don't match the context.

The heatmaps in Figure 4.3 visualize the overlap in error patterns between different models across error categories and datasets. The overlap matrices reveal distinct patterns of agreement and disagreement between models when making errors, with darker colors indicating higher overlap percentages. For *DBpedia*, we observe moderate to high overlap (45-75%) between models across most error categories, suggesting similar challenges in handling complex factual relationships. The *FactBench* dataset shows lower overlap percentages (30-40% typical), indicating more independent error patterns between models. *YAGO*

Table 4.8: Dataset-wise error clustering based on LLM-generated reasoning, using Contextual Document Embeddings for embeddings, UMAP, and HDBSCAN.

Dataset	Model	UnLabeled	Relationship	Role Errors	Geo Errors	Classification	Identifiers	Total*
FactBench	Gemma2	4	36	45	176	13	1	275
	Qwen2.5	33	27	60	194	34	1	349
	Llama3.1	38	44	73	295	38	3	491
	Mistral	53	27	53	242	40	2	417
	Unique. Ratio (%)	0.62	0.72	0.44	0.52	0.63	0.57	0.53
YAGO	Gemma2	6	134	0	14	51	2	207
	Qwen2.5	7	109	0	13	63	2	194
	Llama3.1	8	98	0	19	104	2	231
	Mistral	7	54	0	10	34	3	108
	Unique. Ratio (%)	0.35	0.52	–	0.46	0.51	0.33	0.50
DBpedia	Gemma2	353	22	98	1729	459	299	2960
	Qwen2.5	339	19	91	1525	357	237	2568
	Llama3.1	382	28	109	2172	509	318	3518
	Mistral	325	20	94	1487	438	241	2605
	Unique. Ratio (%)	0.41	0.43	0.44	0.42	0.42	0.40	0.41

* Some errors may not be included in this analysis because we did not receive any responses for them. While we classify these as incorrect predictions, they are not considered in the error analysis section.

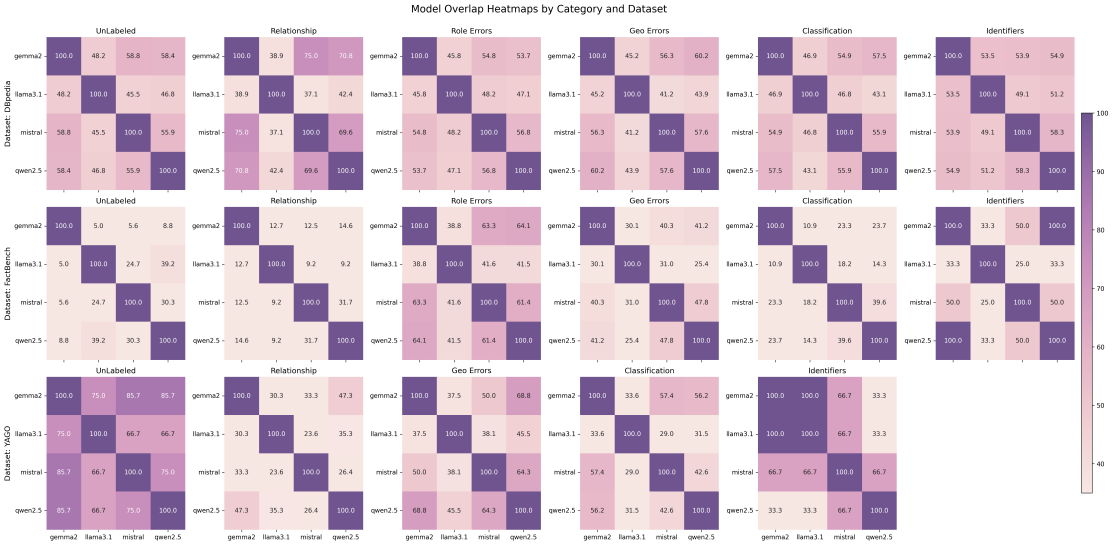


Figure 4.3: Model Overlap Heatmaps by Category and Dataset. Each cell shows the percentage overlap in errors between model pairs. Matrices are organized by error category (UnLabeled, Relationship, Role Errors, etc.) and dataset (DBpedia, FactBench, YAGO), revealing patterns in how models agree or disagree when making verification errors."

exhibits variable overlap, with particularly high agreement in unlabeled errors (75-85% overlap) but lower overlap in relationship errors (23-35%). These patterns suggest that while models often struggle with similar types of facts, they also make distinct errors, supporting the value of ensemble approaches. The lower overlap in *FactBench* errors particularly validates our multi-model verification strategy.

Figure 4.4: Normalized distribution of error clusters across datasets.

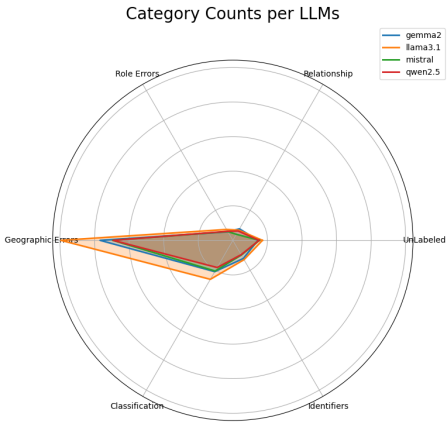
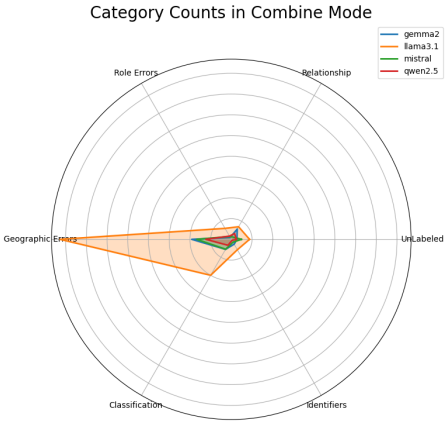


Figure 4.5: Distribution of error clusters across selected LLMs.



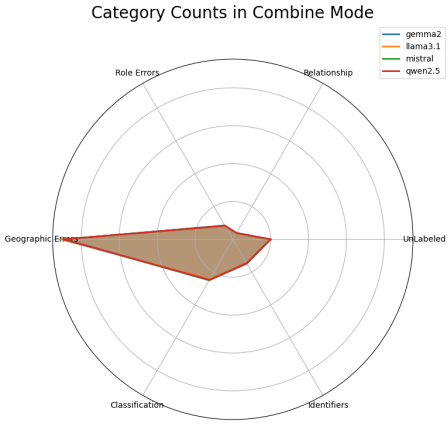


Figure 4.6: Distribution of tendency to be wrong across gemma2, qwen2.5, *LLama3.1* and mistral models. The right chart illustrates the distribution of fully incorrect predictions (4/4) detailing the instances where all predictions made by the models were incorrect. The left chart depicts the distribution of just one wrong predictions (1/4).

Table 4.8 alongside Figures 4.4, 4.5, and 4.6 reveals significant insights into the challenges these models face in fact validation tasks. The study encompassed four leading models *Gemma2*, *LLama3.1*, *Mistral*, and *Qwen2.5* evaluated across three distinct datasets: *FactBench*, *YAGO*, and *DBpedia*. The results highlight both common challenges and model-specific characteristics in fact validation performance.

Geographic and nationality-related errors emerged as the predominant challenge, accounting for 56.9% of total errors across all models and datasets. This pattern was particularly pronounced in the *DBpedia* dataset, where geographic

errors constituted 58.5% of all errors, suggesting a systematic challenge in processing and validating location-based information. This pervasive difficulty across all models indicates a fundamental challenge in handling geographic relationships and facts.

The analysis of dataset-specific patterns revealed distinct characteristics and challenges. The *DBpedia* dataset proved to be the most challenging, generating the highest error count and showing particular vulnerability to geographic and classification errors. In contrast, the *FactBench* dataset demonstrated a more balanced distribution of errors across categories, though still showing a predominance of geographic errors. The *YAGO* dataset exhibited a unique pattern, with relationship errors being the most frequent, followed by classification errors, and notably showing no role attribution errors a distinctive characteristic that sets it apart from other datasets.

When examining model-specific performance, *LLama3.1* consistently generated the highest error counts across datasets, showing particular vulnerability to geographic errors in the *DBpedia* dataset and elevated classification errors compared to other models. *Mistral*, on the other hand, demonstrated stronger overall performance, particularly in the *YAGO* dataset. *Gemma2* and *Qwen2.5* showed similar error patterns and counts, positioning themselves between *LLama3.1* and *Mistral* in terms of performance.

The hierarchical distribution of error types shows a consistent pattern across models. This consistency in error distribution suggests that these challenges are inherent to the task rather than model-specific limitations.

Despite varying error counts, models maintained similar error distribution patterns within each dataset, indicating that these challenges are systematic rather than model-specific. The analysis suggests several critical areas for future development in LLM fact validation capabilities. Primary attention should be directed toward enhancing geographic and location-based reasoning capabilities, given their dominant role in error generation. Additionally, improving classification tasks and the handling of insufficient context or ambiguous information could significantly enhance overall performance. The consistent patterns across models suggest that these improvements would benefit the field broadly rather than being model-specific enhancements.

Building on our previous analysis of *DBpedia* results, we conducted a stratum-wise error analysis to better understand how error distributions vary across different data partitions. Our analysis of error patterns across knowledge strata

showed in Figure 4.7 declares distinct performance characteristics among the four LLMs. The models demonstrated varying levels of effectiveness in handling knowledge from different popularity strata, with error rates showing notable patterns across the commonality spectrum.

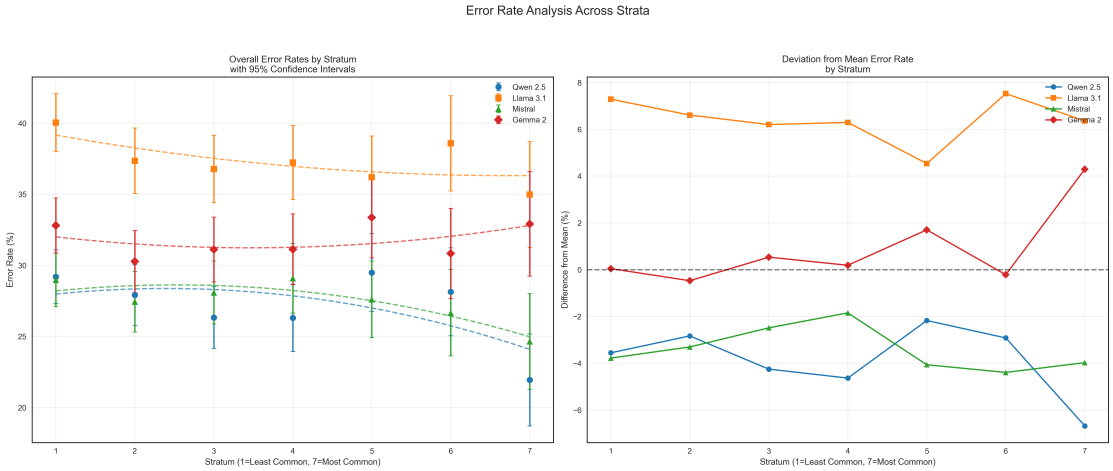


Figure 4.7: Error Distribution Analysis Across Different Language Models and Frequency Strata.

Llama3.1 exhibited the highest overall error rate (37.31% ± 1.51%), significantly exceeding other models' error rates. Despite its higher error rate, *Llama3.1* maintained relatively consistent performance across strata (range: 34.98% - 40.04%), suggesting uniform handling of both common and rare knowledge. The model showed a slight negative correlation with stratum number (r=-0.628, p=0.1309), indicating a modest tendency to perform better with more common knowledge, though this trend was not statistically significant.

In contrast, *Qwen2.5* and *Mistral* demonstrated notably lower error rates (27.04% ± 2.38% and 27.50% ± 1.41% respectively), with *Mistral* showing the most pronounced negative correlation with stratum number (r=-0.761, p=0.0470). This statistically significant correlation indicates that *Mistral's* performance improves substantially as knowledge becomes more common. *Qwen2.5* showed the widest range of error rates (21.94% - 29.49%), suggesting more variable performance across different knowledge types.

Gemma2 maintained an intermediate position with a mean error rate of 31.77% ± 1.13% and showed the most stable performance across strata (range: 30.27% - 33.36%). Uniquely among the models, *Gemma2* exhibited a slight positive correlation with stratum number (r=0.237, p=0.6083), though this trend was not statistically significant. In general, it has the most uniform error distribution

across strata.

Figure 4.8: Distribution of Error Categories Across Different Language Models and Frequency Strata

Analysis of error categories in Figure 4.8 reported separate patterns across strata, with certain error types becoming more prevalent in specific knowledge domains. The proportion of unLabeled errors increased notably in the most common knowledge strata (6-7), while geographic errors showed higher prevalence in less common knowledge strata (1-3). Classification errors maintained relatively consistent proportions across all strata, suggesting that this type of error is less influenced by knowledge commonality.

These findings suggest that while newer models like *Qwen2.5* achieve lower overall error rates, they may be more sensitive to knowledge popularity, performing notably better with common knowledge. In contrast, models like *Gemma2* offer more consistent performance across knowledge types, potentially making them more reliable for applications requiring uniform handling of both common and rare knowledge.

5

Ablation Study

Our proposed framework for knowledge graph fact verification utilizes a unique combination of web search and language model processing. However, to ensure the robustness and effectiveness of our approach, it is crucial to compare our methods with state-of-the-art RAG techniques, particularly in the critical areas of chunking, embedding, and retrieval.

This section aims to provide a comprehensive comparison between our approach and the RAG-based methods. We will focus on four key components of our framework: 1) the retrieval mechanisms utilized to fetch relevant information, 2) the chunking strategies used to segment information, 3) the embedding models employed for representation, and 4) different hyper parameters and configurations. By analyzing these components in light of RAG recommendations, we aim to identify potential areas for improvement and validate the strengths of our current approach.

Through this comparison, we seek to situate our work within the broader context of retrieval-augmented fact verification systems and provide insights into the trade-offs and benefits of our methodological choices. This analysis will not only contribute to the refinement of our framework but also offer valuable perspectives on the application of RAG principles to knowledge graph fact verification tasks.

5.1 Evaluation Methodology

This study employs a systematic approach to evaluate and optimize various components of our framework, with the ultimate goal of determining the best methods for each section. Our methodology is designed to isolate and assess the impact of different techniques and parameters on overall system performance. For our ablation study, we focus specifically on the *FactBench* dataset.

5.1.1 Iterative Optimization Process

The evaluation process follows an iterative strategy, focusing on specific sections of the framework in each iteration:

1. **Section Isolation:** In each iteration, we isolate a particular section of the framework for investigation, keeping other components constant. This "enclosed box" approach allows for a controlled examination of individual elements.
2. **Parameter Variation:** Within the isolated section, we systematically vary relevant parameters or methods.
3. **Performance Evaluation:** For each configuration, we assess the system's performance using predefined metrics (detailed in Sections 5.1.2 and 4.3.1).
4. **Best Method Selection:** Based on the evaluation results, we identify the best-performing method or configuration for the section under investigation.
5. **Incremental Optimization:** The optimal configuration from each iteration is incorporated into the framework for subsequent iterations, gradually refining the entire system.

5.1.2 Evaluation Metrics

The performance of each configuration is assessed using the following metrics, it's same as the metrics in the empirical evaluation section 4.3.1:

- **Accuracy (Acc):** Measures the overall correctness of predictions:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5.1}$$

  where TP, TN, FP, and FN are True Positives, True Negatives, False Positives, and False Negatives, respectively.

- **F1 Score:** Provides a balanced measure of precision and recall:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.2}$$

- **Average Time:** Measured in seconds per query to assess computational efficiency, measured on the Macbook Pro with M2 Max chip and 32GB of RAM.

$$\text{Avg Latency} = \frac{\text{Total Processing Time}}{\text{Number of Queries}} \tag{5.3}$$

5.1.3 Significance of the Methodology

This methodical approach serves several key purposes:

1. **Optimization of Individual Components:** By isolating sections, we can fine-tune each part of the framework independently.
2. **Holistic System Improvement:** The iterative process ensures that optimizations in one section complement the overall system performance.
3. **Efficiency-Accuracy Trade-off Analysis:** Comparing sampling methods to full data runs helps balance computational efficiency with result accuracy.
4. **Scalability Assessment:** This approach informs decisions on system scalability as data volumes increase.

By employing this rigorous evaluation methodology, we aim to identify the best methods for each section of our framework, potentially enabling more efficient and accurate data processing. The inclusion of sampling method comparisons adds an extra dimension to our optimization efforts, potentially offering insights into cost-effective alternatives to full data processing where applicable.

5.2 Document Selection

We explore various techniques for retrieving relevant documents from search engine results, with a specific focus on Google search engine. The goal is to identify the most effective methods for finding documents that perfectly match the information need expressed in the query. We consider both unsupervised and supervised approaches. Using these methods, we aim to find the most relevant documents from the data pool we have collected through web scraping 3.3.3.

5.2.1 Unsupervised Methods

BM25

BM25 [25] is a widely used unsupervised retrieval method that relies on term frequency and inverse document frequency (TF-IDF) weighting. It estimates the relevance of documents to a query based on the frequency of query terms in each document, offset by the rarity of those terms across the full document collection. BM25 has proven to be a robust baseline for many retrieval tasks. However, it relies on lexical matching between query and document terms, which can limit its effectiveness for queries and documents that use different vocabulary to express similar concepts.

$$\text{BM25}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}$$

where:

D : document

Q : query containing keywords $q_1, ..., q_n$

$f(q_i, D)$: frequency of q_i in D

$|D|$: length of document D

avgdl : average document length in the corpus

k_1, b : free parameters

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

where:

N : total number of documents in the corpus

$n(q_i)$: number of documents containing q_i

Contriever

Contriever is a more recently proposed unsupervised method by Izacard et al.[11] that leverages contrastive learning to train dense retrieval models. Rather than relying on term matching, Contriever learns to map semantically similar text pairs to nearby embeddings in a continuous vector space. At query time,

Contriever embeds the query and retrieves the documents whose embeddings are nearest to the query under cosine similarity. By operating in this learned semantic space, Contriever can potentially identify relevant documents that use different surface forms than the query. Contriever has shown promising results, outperforming BM25 on a range of benchmarks when large unsupervised pretraining datasets are available. However, details on its performance in this specific multi-query retrieval setup are needed to fully assess its capabilities here.

While Contriever can be used as an unsupervised retriever, for our thesis project focusing on search-related data, we opt to use the *MS-MARCO* fine-tuned version. [1] Here's why:

- **Relevance to Search Tasks:** MS-MARCO (Microsoft Machine Reading Comprehension) is a large-scale dataset specifically designed for search and question-answering tasks. It contains real queries from Bing search engine and human-annotated relevant passages. By fine-tuning Contriever on MS-MARCO, the model becomes particularly adept at understanding and representing search-like queries and documents.
- **Improved Performance:** Fine-tuning on MS-MARCO significantly boosts Contriever's performance on various retrieval benchmarks, especially those related to web search and question answering. This improvement is crucial for our project, which deals with search-term related data.
- **Domain Adaptation:** Although Contriever's unsupervised training on Wikipedia and CCNet provides a strong foundation, fine-tuning on MS-MARCO helps adapt the model to the specific nuances and patterns present in search queries and web documents. This domain adaptation is valuable for our search-centric application.

5.2.2 Supervised Methods

Jina.ai Reranker

The Jina.ai Reranker is a supervised neural ranking model. Jina Reranker employs a cross-encoder architecture, which represents a paradigm shift from traditional bi-encoder models used in embedding-based search. While bi-encoder models separately encode queries and documents, cross-encoders jointly process query-document pairs, allowing for more nuanced semantic understanding

[1] `https://huggingface.co/facebook/contriever-msmarco`

and relevance assessment. The model generates a relevance score for each query-document pair, enabling a more precise ranking of search results. This approach addresses limitations of vector similarity-based methods by capturing complex token-level interactions between queries and documents.

For our project, we use *jina-reranker-v2-base-multilingual*[2]. This model has demonstrated exceptional performance across various benchmarks and practical applications. In multilingual tasks, it achieved state-of-the-art recall@10 scores on the MKQA dataset [18] spanning 26 languages, while also exhibiting superior NDCG@10 scores on English-language tasks in the BEIR benchmark [33]. Notably, it secured the top position on the AirBench leaderboard upon its release [3].

These capabilities make the model particularly valuable for multilingual information retrieval, agentic RAG systems, and even in programming and software development support.

MS MARCO MiniLM

The *MS MARCO MiniLM* is another supervised neural model, based on the popular BERT architecture trained on the MS MARCO dataset but distilled to a smaller size for efficiency. This comparison in Table 5.1 highlights the trade-

Table 5.1: Performance comparison of various distilled MS MARCO models based on BERT architecture, measured across NDCG@10 on TREC DL 2019 and MRR@10 on MS MARCO Dev benchmarks.

Model Name	**NDCG@10** (TREC DL 19)	**MRR@10** (MS Marco Dev)	**Docs / Sec**
ms-marco-TinyBERT-L-2-v2	69.84	32.56	9000
ms-marco-MiniLM-L-2-v2	71.01	34.85	4100
ms-marco-MiniLM-L-4-v2	73.04	37.70	2500
ms-marco-MiniLM-L-6-v2	74.30	39.01	1800
ms-marco-MiniLM-L-12-v2	74.31	39.02	960

offs between model size, retrieval effectiveness, and processing speed (documents per second). As model size increases from *TinyBERT-L-2-v2* to *MiniLM-L-12-v2*, there is a noticeable improvement in retrieval metrics (NDCG@10 and

[2] `https://huggingface.co/jinaai/jina-reranker-v2-base-multilingual`
[3] `https://huggingface.co/spaces/AIR-Bench/leaderboard`

MRR@10), indicating higher relevance in retrieved documents. However, this comes at the cost of reduced inference speed, with larger models processing fewer documents per second. For our project, we use *ms-marco-MiniLM-L-6-v2*[4] Cross-Encoder model. This model, trained on the extensive MS MARCO dataset comprising approximately 500,000 authentic search queries from the Bing search engine [24], demonstrates superior performance within a two-stage Retrieve & Re-rank framework. In this paradigm, an initial retrieval phase employs either lexical search methods or dense retrieval techniques utilizing a bi-encoder to identify a broad set of potentially relevant documents. Subsequently, the Cross-Encoder refines this candidate set through a simultaneous processing of the query and each retrieved document, generating a relevance score on a scale of 0 to 1.

5.2.3 Evaluation with Large Language Models

Figure 5.1 visualizes similarity patterns in document selection across models, using Jaccard Similarity to quantify the overlap in documents identified as relevant by different models. Higher values indicate greater agreement between models in identifying similar documents, providing insights into consistency and variations in retrieval behavior across the models under evaluation. We

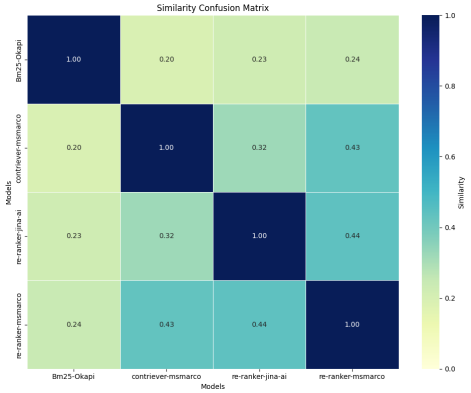


Figure 5.1: Document Retrieval Confusion Matrix based on Jaccard Similarity between documents retrieved by each model.

can figure out that Bm25-Okapi stands out as the most distinct model, with low similarity scores (0.20-0.24) to the others, suggesting it employs fundamentally

[4] `https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2`

different retrieval mechanisms. In contrast, the neural models show higher inter-model similarities, indicating shared approaches or architectures. The strong relationship (0.43) between *contriever-msmarco* and *re-ranker-msmarco*, likely due to shared training data or similar optimizations. The two re-ranker models gain the highest similarity (0.44), highlighting more consistent retrieval patterns. However, they still exhibit differences in document selection. In general, we can find out with different methods we have different result over the same query. To assess the quality of the retrieved documents from each of the above methods, we are passing them through one of our models and evaluating the outputs.

Table 5.2: Performance evaluation of various document retrieval methods on the Fact-Bench dataset, using the Gemma2 model.

	Retrieval Method		
Method	Acc	F1	Latency*
Unsupervised			
Bm25	0.8882	0.8940	**0.4614s**
contriever-msmarco	0.8932	0.8988	25.903s
supervised			
jina-reranker-v2-base-multilingual	0.9004	0.9065	9.8958s
ms-marco-MiniLM-L-6-v2	**0.9014**	**0.9077**	0.8172s

* It is measured in seconds on average per query.

The empirical results indicate that the model *ms-marco-MiniLM-L-6-v2* achieved the highest F1 score, thus demonstrating superior performance among the evaluated models. However, it is noteworthy that the performance metrics across all models were closely clustered, suggesting that even traditional methodologies applied within our pipeline yield satisfactory outcomes.

It is crucial to emphasize the significance of data quality in this context, as it substantially influences the efficacy of the results. To validate the factual accuracy of the knowledge graph, we employed a multi-query information fetching through web search engines for each fact. This approach provides a reasonable degree of verification for the facts contained within the knowledge graph.

For subsequent evaluations and analyzes, we will designate the model *ms-marco-MiniLM-L-6-v2* as our baseline for retrieval tasks. This decision is predicated on its superior accuracy and F1 score relative to the other models under consideration with acceptable latency.

5.3 Embedding Models

Text embeddings are dense vector representations that capture the semantic meaning and relationships between words, sentences, or documents in a low-dimensional space. By mapping text to a continuous vector space, embeddings enable efficient similarity computations and have become a fundamental building block for many NLP applications, such as information retrieval, text classification, clustering, and semantic search. This section provides an in-depth analysis and comparison of five state-of-the-art text embedding models:

- Alibaba-NLP/gte-large-en-v1.5
- jinaai/jina-embeddings-v3
- dunzhang/stella_en_1.5B_v5
- Nextcloud-AI/multilingual-e5-large-instruct
- BAAI/bge-small-en-v1.5

These models leverage recent advancements in transformer architectures, contrastive learning, and instruction fine-tuning to produce high-quality, general-purpose embeddings that excel across a wide range of downstream tasks. We examine their model architectures, training methodologies, supported features, and empirical performance on standard benchmarks. Through this comparative study, we aim to provide insights and guidance for practitioners to select the most suitable embedding model based on their specific use case and computational constraints.

5.3.1 Gte-large-en-v1.5

The Alibaba-NLP model *gte-large-en-v1.5* is text embedding model designed for general text representation and retrieval tasks. It is built upon a Transformer++ encoder architecture, combining the strengths of BERT [4] with advanced techniques such as rotary position embeddings (RoPE) [29] and Gated Linear Units (GLU). This combination allows for highly efficient text encoding over long sequences, with a maximum context length of 8192 tokens, significantly surpassing previous models restricted to shorter context lengths (up to 512 tokens) [39].

One of the major improvements in the gte-v1.5 series is its ability to process long-context text inputs, making it ideal for complex text retrieval and

re-ranking tasks [16]. This series of models has demonstrated superior performance in multiple benchmarks, including the Massive Text Embedding Benchmark (MTEB) [22] and the LoCo long-context retrieval benchmark [26]. In particular, the tuned models of this model ranked second on the MTEB leaderboard and first in the Chinese version of MTEB (C-MTEB).

The model achieves these results by employing a hybrid architecture, including both a text representation model (TRM) and a cross-encoder reranker. The TRM generates dense text embeddings for retrieval tasks, while the reranker refines results through more precise scoring of candidate texts. This architecture is optimized for efficiency, allowing faster inference while maintaining high accuracy during both pretraining and fine-tuning stages.

The *gte-large-en-v1.5* also includes instruction-tuned variants, such as *gte-Qwen1.5-7B-instruct*, which is particularly effective for multilingual text embeddings, leveraging a wide range of unsupervised and supervised contrastive learning techniques. These instruction-tuned models have outperformed various other large embedding models, making them highly suitable for industrial applications that require efficient, accurate text representation across diverse languages.

In summary, the *gte-large-en-v1.5* model stands out in its category due to its ability to handle large context lengths, its efficient encoding techniques, and its strong performance on long-context benchmarks. This makes it an invaluable tool for a variety of text retrieval, classification, and representation tasks in both academic research and real-world applications.

5.3.2 Jina-embeddings-v3

The *Jina-embeddings-v3* model is a cutting-edge multilingual text embedding solution, developed by Jina AI, aimed at addressing a wide range of NLP tasks. Based on the *Jina-XLM-RoBERTa* architecture, this model supports long-context inputs, handling sequences of up to 8192 tokens thanks to its integration of RoPE [29, 28].

This ability to process extended sequences makes the model well-suited for tasks such as text retrieval, clustering, classification, and text matching across multiple languages. One of the key innovations of *Jina-embeddings-v3* is the introduction of task-specific Low-Rank Adaptation (LoRA) [10] adapters. These adapters are used to tailor the model's embeddings to specific tasks, such as

query-document retrieval, clustering, re-ranking, and classification. This task-specific optimization is achieved without significantly increasing the model's parameter size.

The model excels in multilingual environments, supporting wide range of languages, and is optimized for performance in long-context retrieval tasks. Compared to LLMs like *e5-mistral-7b-instruct*, *jina-embeddings-v3* offers a more efficient solution with fewer parameters (570 million *vs*. 7.1 billion), while still achieving competitive or superior performance on several benchmarks. For example, it surpasses proprietary models like OpenAI [5] and Cohere [6] on English tasks and achieves high scores on multilingual benchmarks.

Jina-embeddings-v3 also features flexible Matryoshka Representation Learning (MRL) [14], allowing users to reduce the embedding size from 1024 to as low as 16 dimensions, making it adaptable to different resource constraints without significant loss of performance.

5.3.3 Stella_en_1.5B_v5

The Dunzhang *Stella_en_1.5B_v5*[7] is a powerful multilingual text embedding model, built upon the foundations of *Alibaba-NLP/gte-large-en-v1.5* 5.3.1 and *gte-Qwen2-1.5B-instruct*. This model supports two main prompts for diverse tasks: "s2p" (sentence-to-passage) for information retrieval, and "s2s" (sentence-to-sentence) for semantic textual similarity. These prompts simplify its application in NLP tasks, such as retrieving relevant passages or finding semantically similar text based on a given query.

One of the standout features of *Stella_en_1.5B_v5* is its implementation of MRL [14], allowing the model to output embeddings in multiple dimensions ranging from 512 to 8192, depending on user needs. Typically, a 1024-dimensional output offers an optimal balance between performance and efficiency. In benchmark tests, the model achieves highly competitive results, with only a minor performance difference between 1024-dimensional and 8192-dimensional embeddings. The model can be employed using both SentenceTransformers and transformers libraries, supporting flexible input formats. It is trained on shorter

[5] https://openai.com/
[6] https://cohere.com/
[7] https://huggingface.co/dunzhang/stella_en_1.5B_v5

sequences (up to 512 tokens), making it most effective for short-to-medium-length text tasks.

5.3.4 Multilingual-e5-large-instruct

The Multilingual E5-Large-Instruct model is an advanced multilingual text embedding model introduced as part of the E5 model family, which aims to improve the quality and utility of multilingual text embeddings [35]. It is specifically designed to support a wide range of languages and to deliver robust performance across various tasks such as text retrieval, semantic similarity, and multilingual retrieval.

The E5-Large-Instruct model contains 24 layers and features an embedding size of 1024. It builds on the *XLM-RoBERTa-large* [3] architecture, which supports 100 languages, albeit with varying performance depending on the resource richness of the language in question. The model was initialized from XLM-RoBERTa-large and underwent two key stages of training:

- **Contrastive Pre-training:** The model was pre-trained on approximately 1 billion weakly supervised multilingual text pairs using a InfoNCE contrastive loss with only in-batch negatives, while other hyperparameters remain consistent with the English E5 models.

- **Fine-tuning:** Following pre-training, the model was fine-tuned using high-quality labeled datasets from the E5-mistral paper [34]. This second stage involved a more supervised approach, optimizing performance across specific tasks. During this phase, instruction-tuning was incorporated, where the model learned to generate better embeddings by using natural language task instructions.

The E5-Large-Instruct model was evaluated on BEIR and MTEB benchmarks, and its performance is on par with state-of-the-art English-only models. Evaluation on the MIRACL [40] multilingual retrieval benchmark across 16 languages and on Bitext mining tasks across over 100 languages demonstrated its capability to handle diverse languages effectively. Despite the excellent performance on high-resource languages, the model shows a little degradation in performance for low-resource languages, a common limitation of multilingual models, but still outperforms many other models in this category. The use of contrastive learning and instruction tuning enables the model to generate highly effective embeddings for information retrieval tasks.

5.3.5 BGE-SMALL-EN-V1.5

The *bge-small-en-v1.5* model is part of the BGE (BAAI General Embeddings) series developed by the Beijing Academy of Artificial Intelligence [36]. It is a compact English-specific model with just 33.4M parameters, making it highly efficient for deployment in resource-constrained environments. The model architecture is BERT-like which goes through three-stage of training. *bge-small-en-v1.5* follows a two-stage training pipeline similar to other BGE models:

- **Pre-training:** Weakly-supervised contrastive pre-training on large-scale web data
- **Fine-tuning:** Supervised fine-tuning on a curated set of high-quality English NLP datasets

The model fine-tuned using a process of contrastive learning, where sentences are embedded to prioritize semantic similarity. This technique enhances retrieval tasks by training the model to produce high similarity scores for semantically related sentences while keeping unrelated pairs distant in embedding space. The fine-tuning emphasizes retrieval for short queries to long passages, optimized with a contrastive loss function and often utilizes mined hard negatives to improve differentiation between similar and unrelated sentence pairs.

Despite its small size, *bge-small-en-v1.5* punches above its weight on several English benchmarks and outperforms the base-sized BERT and RoBERTa models on most tasks while being more compact. The model's strong performance can be attributed to the efficient architecture design and the use of high-quality fine-tuning data. It presents an attractive option for applications requiring low-latency inference or deployment on edge devices.

5.3.6 Comparative Analysis

We examine their model size and efficiency, language coverage, supported features, and overall performance to provide insights for selecting the most suitable model based on specific requirements.

Model Size and Efficiency

Table 5.3 compares the model size, memory usage, embedding dimensions, and maximum token length of the five models. The *stella_en_1.5B_v5* model has the largest size with 1,543 million parameters, while *bge-small-en-v1.5* is

the smallest with only 33 million parameters. Larger models generally require more memory and computational resources, which may be a consideration for resource-constrained environments. In terms of memory usage, *stella_en_1.5B_v5* requires 5.75 GB in fp32 precision, while *bge-small-en-v1.5* only needs 0.12 GB. This substantial difference in memory footprint can be a decisive factor when deploying models on edge devices or serving them in real-time applications with limited resources. The embedding dimensions also vary among the models, ranging from 384 for *bge-small-en-v1.5* to 8192 for *stella_en_1.5B_v5*. Higher-dimensional embeddings can capture more fine-grained semantic information but may increase storage requirements and similarity computation costs. Practitioners should consider the trade-off between embedding quality and efficiency based on their specific use case.

Table 5.3: Comparison of characteristics of embedding models

Model	**Model Size** (Million Parameters)	**Memory Usage** (GB, fp32)	**Embedding Dimensions**	**Max Tokens**
stella_en_1.5B_v5	1543	5.75	8192	131072
jina-embeddings-v3	572	2.13	1024	8194
gte-large-en-v1.5	434	1.62	1024	8192
multilingual-e5-large-instruct	560	2.09	1024	514
bge-small-en-v1.5	33	0.12	384	51262

Language Coverage

Language coverage is a crucial aspect when selecting an embedding model for multilingual applications. The *Multilingual-e5-large-instruct* model stands out in this regard, as it supports a wide range of languages. This model leverages instruction fine-tuning on multilingual data, enabling it to generate high-quality embeddings for various languages. The *Jina-embeddings-v3* model also offers multilingual support, although the exact language coverage is not specified in the provided context. On the other hand, the *Bge-small-en-v1.5*, *Stella_en_1.5B_v5*, and *Gte-large-en-v1.5* models primarily focus on English embeddings, making them more suitable for monolingual English applications.

Conclusion

The choice of text embedding model depends on various factors, including the specific application, language coverage requirements, available computa-

tional resources, and desired features. For monolingual English applications, the *Gte-large-en-v1.5* and *Stella_en_1.5B_v5* models offer high-quality embeddings with support for longer input sequences. The *Stella_en_1.5B_v5* model, in particular, provides prompt-based adaptability for information retrieval and semantic similarity tasks. For multilingual applications, the *Multilingual-e5-large-instruct* and *Jina-embeddings-v3* models are strong contenders. The *Multilingual-e5-large-instruct* model supports a wide range of languages, while *Jina-embeddings-v3* offers task-specific LoRA adapters for enhanced performance across various NLP tasks. When computational resources are limited, the *Bge-small-en-v1.5* model presents a lightweight option with competitive performance. Its small size and low memory footprint make it suitable for deployment on edge devices or real-time applications. Ultimately, we should carefully evaluate our specific requirements and constraints before selecting an embedding model. The comparative analysis provided in this section aims to assist in this decision-making process by highlighting the key differences and strengths of each model. Now we will test these models through the pipeline and evaluate their performance.

Table 5.4: Performance evaluation of various embedding models on the FactBench dataset, using the Gemma2 model.

ms-marco-MiniLM-L-6-v2, *Embedding Model*

Model	Acc	F1	Latency
stella_en_1.5B_v5	0.8961	0.9028	17.692s
multilingual-e5-large-instruct	0.8954	0.9018	5.0038s
bge-small-en-v1.5	**0.9014**	0.9077	**1.6958s**
jina-embeddings-v3*	0.8852	0.9097	4.8745s
gte-large-en-v1.5*	0.8971	**0.9174**	5.8571s

* The models were not able to complete the evaluation due to memory constraints, Jina evaluated on the 2238/2800 and Gte-large evaluated on the 2322/2800.

Based on the Table 5.4, we use the *bge-small-en-v1.5* model for the subsequent evaluations and analyses due to its superior performance across F1 and accuracy metrics. The low latency of 1.6958 seconds per query also makes it an attractive choice for real-time applications. The F1 score of *gte-large-en-v1.5* is slightly higher, but the model is not able to complete the evaluation due to memory limitations in the same pipeline.

5.4 Chunking Strategies

A critical component of RAG systems is the chunking strategy employed to divide documents into smaller, manageable pieces for efficient retrieval and processing. This section examines three distinct chunking methods for RAG systems, each with its unique characteristics and potential advantages.

5.4.1 Parsing Documents into Text Chunks

The first method we will explore involves parsing documents into text chunks, also referred to as nodes, of fixed sizes. This approach is straightforward and widely used in many RAG implementations. We will investigate three different chunk sizes: 256, 512, and 1024 tokens.

Methodology

In this method, documents are sequentially divided into chunks of the specified size. If the final chunk is smaller than the designated size, it is typically padded or left as is, depending on the implementation.

Chunk Sizes

- **256-token chunks:** This size offers fine granularity, potentially allowing for more precise retrieval of relevant information. However, it may result in a loss of context for more complex topics that require broader context.
- **512-token chunks:** This medium-sized chunk strikes a balance between granularity and context preservation. It is often considered a good default choice for many applications.
- **1024-token chunks:** Larger chunks preserve more context but may retrieve more irrelevant information and increase computational overhead during retrieval and processing.

5.4.2 Smaller Child Chunks Referring to Bigger Parent Chunks (Small2Big)

The second method, which we will refer to as *Small2Big*, involves creating a hierarchical structure of chunks, where smaller child chunks refer to larger

parent chunks. This approach aims to combine the benefits of fine-grained retrieval with the context preservation of larger chunks.

Methodology

In this method, we parsed documents into three levels of chunks with appending the original text chunk of size 1024:

- Smallest children: 128-token chunks
- Intermediate parents: 256-token chunks
- Largest parents: 512-token chunks

Each smaller chunk maintains a reference to its parent chunks, allowing the system to retrieve additional context when needed.

```
# ...previous code
sub_chunk_sizes = [128, 256, 512]
sub_node_parsers = [SimpleNodeParser.from_defaults(chunk_size=c) for c in sub_chunk_sizes]

all_nodes = []
for base_node in base_nodes:
    for n in sub_node_parsers:
        sub_nodes = n.get_nodes_from_documents([base_node])
        sub_inodes = [
            IndexNode.from_text_node(sn, base_node.node_id) for sn in sub_nodes
        ]
        all_nodes.extend(sub_inodes)

    original_node = IndexNode.from_text_node(base_node, base_node.node_id) # also add original node to node
    all_nodes.append(original_node)
all_nodes_dict = {n.node_id: n for n in all_nodes}
# ... continue processing
```

Code 5.1: Small2Big Chunking Method

5.4.3 Sentence Window Retrieval

The third method, Sentence Window Retrieval, focuses on maintaining semantic coherence by chunking based on sentences and incorporating surrounding context through windows.

Methodology

In this approach, documents are first split into individual sentences. For each sentence, a *window* of surrounding sentences is included to provide context. we use the *SentenceWindowNodeParser* to parse documents into single sentences per node. Each node also contains a *"window"* with the sentences on either side

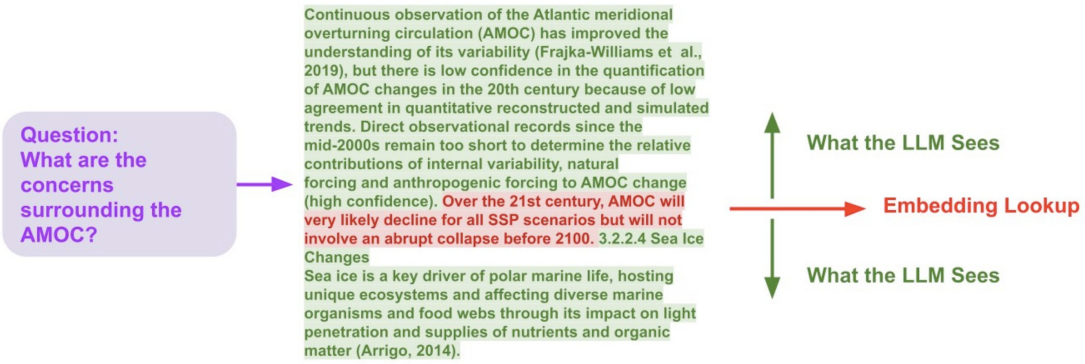


Figure 5.2: Node sentence window replacement technique as described by Liu [17].

of the node sentence. Then, after retrieval, before passing the retrieved sentences to the LLM, the single sentences are replaced with a window containing the surrounding sentences using the MetadataReplacementNodePostProcessor. This is most useful for large documents/indexes, as it helps to retrieve more fine-grained details. We will examine two window sizes: 3 and 6.

5.4.4 Advantages and Limitations

Table 5.5 provides an analysis of the advantages and limitations of each text segmentation method based on their methodological approach. Each of the three chunking methods presented in this section offers distinct advantages and limitations for RAG systems. The choice of method depends on factors such as the nature of the documents, the specific requirements of the application, and the computational resources available. The fixed-size chunking method provides simplicity and consistency but may sacrifice semantic coherence. The Small2Big hierarchical approach offers flexibility in retrieval granularity but introduces complexity in implementation and storage. Sentence Window Retrieval preserves semantic units and adapts to text structure but may result in variable chunk sizes.

Examples of the three chunking methods are available in the Appendix B for further reference.

Table 5.5: Advantages and Limitations of different chunking strategies for RAG systems.

Method	Advantages	Limitations
Fixed Chunking	* Simple to implement and understand * Consistent chunk sizes facilitate uniform processing	* Fixed chunk sizes may not align with natural breaks in the text * Larger chunks can introduce irrelevant information and increase computational costs
Small2Big	* Allows for fine-grained retrieval with the option to expand context * Adapts to different levels of specificity required by queries	* More complex to implement and manage * Increased storage requirements due to redundancy in the hierarchy
Sentence Window	* Preserves semantic units (sentences) and their immediate context * Adapts to the natural structure of the text	* Variable chunk sizes may complicate processing and indexing * Optimal window size may vary depending on the document type and content

5.4.5 Evaluation

The Table 5.6 illustrates that as the chunk size increases, there is a minor downtick in the average latency. Consider that the pipeline's average response time increases as the chunk size increases, which is expected as the model has to process more tokens. Interestingly, the faithfulness (*i.e.* measuring how closely response is aligned with the source material) seems to reach its zenith at chunk_size of 1024, whereas average relevancy shows a consistent improvement with larger chunk sizes, also peaking at 1024. This suggests that a chunk size of 1024 might strike an optimal balance between response time and the quality of the responses, measured in terms of faithfulness and relevancy. In the sliding window method, the window size of 3 outperforms the window size of 6 in terms of both accuracy and F1 score, while maintaining nearly the same average response time. We can figure out that the baseline accuracy and F1 scores across chunking strategies indicate that the model's performance is largely unaffected by these variations, suggesting that factors other than chunk size, may have a more significant impact on the retrieval process.

Table 5.6: Performance evaluation of various chunking strategy on the FactBench dataset, using the Gemma2 model.

ms-marco-MiniLM-L-6-v2, BAAI/bge-small-en-v1.5, *Chunking Strategy*

Method	**Parameters**	Acc	F1	Latency
Original	Chuck Size: 256	0.8914	0.8914	0.04473s
	Chuck Size: 512	0.8932	0.8993	0.02670s
	Chuck Size: 1024	0.8946	0.8993	**0.02378s**
small2big	Chuck Size: 1024	0.8889	0.8953	0.19188s
Sliding Window	Window Size: 3	**0.9014**	**0.9080**	0.03076s
	Window Size: 6	**0.9014**	0.9077	0.03534s

We chose the Sliding Window with window size 3 method as it provides the surrounding context of the text and has the highest F1 score and accuracy.

5.5 Similarity Cut-off

In this section we will discuss how similarity cut-off can be used to filter out irrelevant nodes and improve the efficiency of the retrieval process. We use Node postprocessors to apply a similarity cut-off to the retrieved nodes, discarding those with a similarity score below a certain threshold. Node postprocessors are a set of modules that take a set of nodes, and apply some kind of transformation or filtering before returning them. For our experiments, we set the similarity cut-off threshold to 0.3, meaning that nodes with a similarity score below 0.3 are discarded, we use the naive score and the re-ranker score to compare the results. The Algorithm 3 shows the similarity cut-off postprocessor implementation.

Algorithm 3 Similarity Cutoff Postprocessor (re-rank score)

```
procedure PostprocessNodes(nodes, knowledge_graph, similarity_cutoff)
    new_nodes ← []
    node_texts ← [node.text for node in nodes]
    re_rank_nodes ← ReRank(knowledge_graph, node_texts)
    for each node in nodes do
        node.score ← get_node_score(node.text, re_rank_nodes)
        if node.score > similarity_cutoff then
            new_nodes.Append(node)
        end if
    end for
    return new_nodes
end procedure
```

Table 5.7: Performance evaluation of similarity cut-off method on the FactBench dataset, using the Gemma2 model.

ms-marco-MiniLM-L-6-v2, BAAI/bge-small-en-v1.5, Sliding Window (ws 3), *Similarity Cut-off*

Method	Acc	F1	Latency Diff.*
w/o similarity cut-off (baseline)	0.8971	0.9036	–
similarity cut-off (original score)	**0.9018**	**0.9080**	-0.22237s
similarity cut-off (re-ranked score)	0.9014	**0.9080**	-0.350783

* The latency is compared to the baseline without the similarity cut-off on average per query.

Based on the results in Table 5.7, we decided to apply a similarity cut-off with the original score to provide the model with higher-quality data for further evaluations. This approach yields the highest accuracy and F1 score, though it is slightly slower than the re-ranker mode because it removes fewer irrelevant nodes. A drawback of re-ranking is that it may eliminate all relevant nodes based on low similarity score, requiring a re-run without the similarity cut-off, which increases processing time (not shown in the Table 5.7). In separate evaluations (not reported in Table 5.7), we also tested a normalized similarity cut-off using re-ranked scores scaled between the original minimum and maximum values. However, this normalized approach did not perform as well as the original scores.

5.6 Top K

Top_k mentions how many top embeddings to take into context. Considering a large top_k might go beyond the max_tokens of the model, we will evaluate the performance of the pipeline with the top_k set to 3 and 6, and compare the results to determine the optimal value for this parameter.

Table 5.8: Performance evaluation of different Top_k retrieval strategies on the FactBench dataset using the Gemma2 model.

ms-marco-MiniLM-L-6-v2, BAAI/bge-small-en-v1.5, Sliding Window (ws 3), Similarity Cut-off (Original), *Top_k*

Method	Acc	F1	Latency*
Top_k 3	0.9018	0.9080	**5.21177s**
Top_k 6	**0.9032**	**0.9101**	7.02713s

* The latency represents the average time per query for a complete run.

As we decided to set the similarity cut-off with 0.3 threshold, it's good to use the top_k 6 to have more high-quality embeddings in the context, and based on results on Table 5.8 it outperforms the top_k 3, so we will use top_k 6 for subsequent evaluations. The difference in latency is significant, presenting a trade-off between more data and response time. Since the quality of data for additional facts is uncertain, we aim to include more data in the context.

5.7 Evaluation

In this ablation study, we evaluate the performance of the merging method, which leverages a novel ensemble approach by combining multiple models (*Gemma2*, *Qwen2.5*, *Llama3.1*, and *Mistral*) to show the robustness and accuracy of the pipeline. As presented in Tables 5.9 and 5.10, along with Figure 5.3, the proposed ensemble method consistently outperforms individual models across both positive and negative labels, achieving a more balanced and comprehensive performance.

Note that the ensemble method is based on the tie-breaking strategy discussed in Section 3.6.2, with *At_Most* as the merging method. Based on the provided configurations the model selected for *At_Most* methodology is the *Gemma2:21B*.

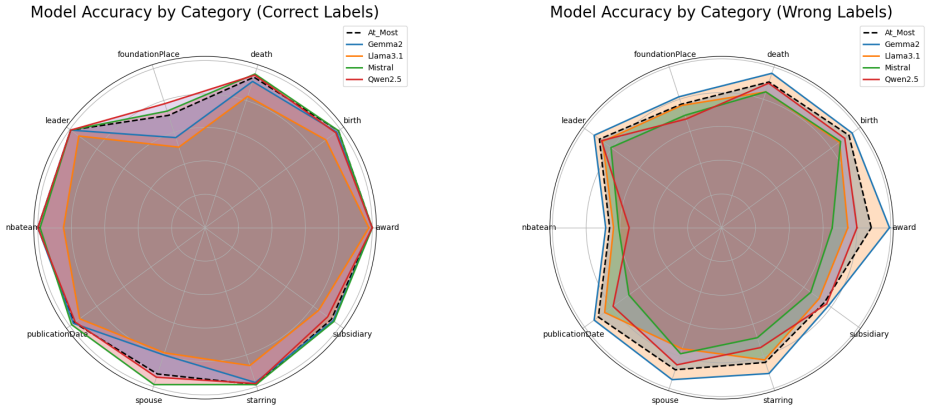


Figure 5.3: Category-wise performance of different models in identifying Positive Labels (left) and Negative Labels (right) on the FactBench dataset.

Table 5.9: Category-wise performance evaluation results of various models on the FactBench dataset.

ms-marco-MiniLM-L-6-v2, BAAI/bge-small-en-v1.5, Sliding Window (ws 3), Similarity Cut-off (Original), Top_k 6

Model	award	birth	death	foundationPlace	leader	nbateam	publicationDate	spouse	starring	subsidiary	Total
Positive Labels											
Gemma2	**1.0000**	0.9733	0.9200	0.5667	**0.9933**	**1.0000**	0.9733	0.8000	0.9733	0.9467	0.9147
Qwen2.5	**1.0000**	0.9667	**0.9600**	**0.7800**	**0.9933**	**1.0000**	0.9600	0.9400	0.9800	0.9067	**0.9487**
Llama3.1	0.9800	0.8933	0.8267	0.5067	**0.9333**	0.8467	0.9267	0.7867	0.8667	0.8400	0.8407
Mistral	**1.0000**	**0.9867**	0.9667	0.7333	**0.9933**	0.9867	**0.9867**	**0.9867**	**0.9867**	**0.9533**	0.9580
Proposed (At_Most)	**1.0000**	**0.9867**	0.9467	0.7067	**0.9933**	**1.0000**	0.9667	0.9200	**0.9867**	0.9333	0.9440
Negative Labels											
Gemma2	**0.9923**	**0.9538**	**0.9615**	**0.8154**	**0.9308**	**0.6846**	**0.9308**	**0.9462**	**0.9077**	**0.7769**	**0.8900**
Qwen2.5	0.8000	0.9000	0.9000	0.6769	0.8769	0.5462	0.7923	0.8538	0.7462	0.7615	0.7854
LLama3.1	0.7462	0.8615	0.8462	0.7615	0.8692	0.6385	0.8538	0.7538	0.8231	0.7077	0.7862
Mistral	0.6538	0.8692	0.8462	0.7000	0.8077	0.6077	0.6769	0.7846	0.6846	0.6462	0.7277
Proposed (At_Most)	0.8846	0.9308	0.9077	0.7692	0.8923	0.6615	0.9000	0.8846	0.8385	0.7462	0.8415

Table 5.10: Performance evaluation of various models on the FactBench dataset.

ms-marco-MiniLM-L-6-v2, BAAI/bge-small-en-v1.5, Sliding Window (ws 3), Similarity Cut-off (Original), Top_k 6

Model	Consistency[a]	Avg. Request Time[b]	Avg. tokens per request[c]	ACC	F1
Gemma2	0.8720	5.6826s	1605.29	**0.9032**	**0.9101**
Qwen2.5	0.8685	6.3094s	1652.66	0.8729	0.8888
LLama3.1	0.8291	6.5270s	1679.04	0.8154	0.8299
Mistral	0.8650	4.6692s	1594.76	0.8511	0.8733
Proposed (At_Most)	**0.9176**	16.815s	1604.196	0.8964	0.9071

[a] Consistency score for each individual model is calculated across all models, excluding the ensemble models.
[b] Each query uses two requests, so the average request duration is calculated based on the two requests.
[c] The average is calculated based on the total number of input and output tokens combined.

5.8 Failure Analysis

To gain a deeper understanding of the limitations and challenges faced by our fact-checking system, we conducted a comprehensive failure analysis using the *FactBench* dataset. By examining the instances where our system, based on the majority vote, failed to correctly verify the facts, we aimed to identify the main error types and provide insights into the reasons behind these failures.

Error Type Categorization

After analyzing the failure cases, we categorized the errors into four main types:

- **Insufficient or Irrelevant Context:** In some cases, the provided context

information does not directly support or refute the given triple. The LLMs struggle to make accurate judgments when the necessary facts are missing or the available information is tangentially related to the claim.

- **Misinterpretation of Relationships:** The LLMs sometimes misinterpret the relationships between entities mentioned in the context. They may confuse family relations, professional associations, or the nature of events.
- **Over reliance on Keyword Matching:** In some instances, the LLMs rely too heavily on surface-level keyword matching rather than understanding the underlying semantics. The presence of certain words or phrases can lead to incorrect assumptions.
- **Lack of Common Sense Reasoning:** The LLMs can struggle with applying common sense knowledge or reasoning about the plausibility of claims. They may fail to consider the unlikelihood of certain scenarios or relationships.

Table 5.11: Example of failure cases and error analysis observed in the FactBench dataset using generated results and explanations.

Error Type	Triple	Description
Insufficient or Irrelevant Context	Ai Sugiyama birth place Yokohama	Since there's no information in any of the documents about Ai Sugiyama being born in Yokohama, and one document explicitly states her birthplace as Tokyo. So LLms infer that Ai Sugiyama was born in Tokyo, Japan and not Yokohama, Japan.
Mis Interpretation of Relationships	Mitt Romney office Dallas	LLMs mistakenly infer that Romney has an office in Dallas based on his attendance at a fundraiser there. Attending an event doesn't imply having a permanent office.
Over reliance on Keyword Matching	Robbie Williams office Los Angeles	LLMs wrongly assume Robbie Williams has an office in Los Angeles due to text discussing his purchase or sell of a property there, not an office.
Lack of Common Sense Reasoning	Saul Bellow starring Nobel Prize in Literature	LLMs fail to recognize "starring" is inappropriate for receiving a Nobel Prize. Common sense suggests terms like "awarded" or "received."

Based on Figure 5.4, The "foundationPlace" relation shows the highest number of total instances and errors in both charts. This suggests that the model struggles most with verifying facts about the locations where organizations or institutions were founded. The large discrepancy between correct and incorrect predictions for this relation indicates a significant challenge in accurately processing location-based information.

Relations such as "birth", "death", and "spouse" show varying levels of difficulty. While "birth" and "death" have relatively few instances, "spouse" has a moderate number of cases with a notable error rate. This suggests that verifying personal information, especially relationships, poses challenges for the model, and mostly related to having multiple marriages or relationships during a lifetime.

The "nbateam" and "subsidiary" relations, which involve organizational affiliations, show moderate error rates. This indicates that the model has some difficulty in correctly identifying professional associations and corporate structures.

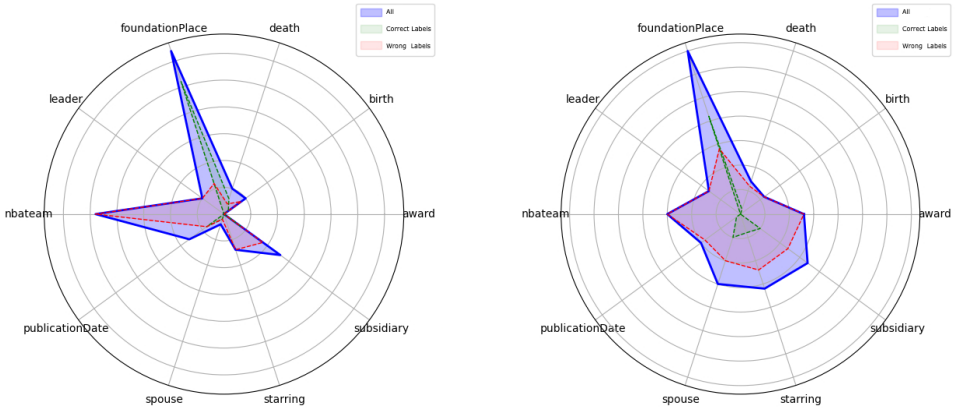


Figure 5.4: Prediction accuracy on the FactBench dataset, focusing on incorrect predictions. The right chart illustrates the Distribution of Fully Incorrect Predictions (4/4), detailing the instances where all predictions made by the models were incorrect. The left chart depicts the Distribution of Partially Incorrect Predictions (3/4).

In general, by spotting Figure 5.4, we can observe that the overall shape of the error distribution is similar, indicating consistency in the model's performance across different voting thresholds.

6

Conclusions and Future Works

This thesis has presented a novel approach to knowledge graph fact verification using RAG. Through extensive experimentation and analysis, we have demonstrated the effectiveness of combining multiple large language models with sophisticated information retrieval techniques to verify facts in knowledge graphs. The key findings and contributions of this work can be summarized as follows:

- **Pipeline Architecture:** We have developed a comprehensive pipeline that integrates web search, document processing, and multiple language models to verify knowledge graph facts. The pipeline's modular design allows for flexibility and future improvements in individual components.
- **Multi-Model Integration:** Our approach of combining multiple language models through majority voting and adaptive dispute resolution has proven effective in improving the overall accuracy and reliability of fact verification. The system achieved an acceptable accuracy and F1 score on the FactBench and Yago datasets, demonstrating its capability to handle diverse fact types.
- **Processing Optimization:** Through ablation studies, we identified optimal configurations for document selection, embedding models, and chunking strategies.
- **Error Analysis:** Our detailed analysis of failure cases has provided valuable insights into the system's limitations and areas requiring improvement, particularly in handling complex relationships and insufficient context scenarios.

Based on our findings and identified limitations, several promising directions for future research emerge:

1. **Enhanced Context Processing:** Develop more sophisticated methods for handling cases with insufficient or irrelevant context. Implement better techniques for identifying and resolving contradictions in retrieved information.
2. **Model Integration:** Explore additional strategies for combining model outputs beyond majority voting. Investigate dynamic model selection based on query characteristics. Implement more sophisticated tie-breaking mechanisms.
3. **Retrieval Optimization:** Improve query generation for better coverage of fact verification requirements. Develop more effective filtering mechanisms for irrelevant information. Enhance the similarity cut-off strategy for more precise document selection.
4. **Scalability Improvements:** Optimize computational resource usage for handling larger knowledge graphs. Develop more efficient document processing and embedding techniques. Implement parallel processing capabilities for faster verification.
5. **Explainability and Transparency:** Develop better methods for explaining verification decisions. Implement confidence scoring mechanisms. Create visualization tools for the verification process.
6. **Domain Adaptation:** Create specialized verification strategies for different types of facts. Develop domain-specific knowledge integration mechanisms. Implement adaptive learning capabilities for new domains.

Prompt Templates

In this section, we present the prompt templates used in the pipeline.

A.1 Human-understandable text generation Prompt

Prompt template for generating human-readable text

```
Task Description:
Convert a kg triple into a meaningful human readable sentence.

Instructions:
    Given a subject, predicate, and object from a kg, form a
    grammatically correct and meaningful sentence that conveys
    the relationship between them.

Examples:
Input:
    Subject: Alexander_III_of_Russia
    Predicate: isMarriedTo
    Object:  Maria_Feodorovna__Dagmar_of_Denmark_
    Output: {"output" : "Alexander III of Russia is married to Maria
                         Feodorovna, also known as Dagmar of Denmark."}

Input:
    Subject: Quentin_Tarantino
    Predicate: produced
```

```
    Object: From_Dusk_till_Dawn
    Output: {"output": "Quentin Tarantino produced the film
                        From Dusk till Dawn."}

Input:
    Subject: Joseph_Heller
    Predicate: created
    Object: Catch-22
    Output: {"output": "Joseph Heller created the novel Catch-22."}

Do the following:
Input:
Subject: {knowledge_graph.subject}
Predicate: {knowledge_graph.predicate}
Object: {knowledge_graph.object}
The output should be a JSON object with the key "output" and
the value as the sentence. The sentence should be human-readable
and grammatically correct. The subject, predicate, and object
can be any valid string without having extra information.
```

A.2 Question Generation Prompt

Prompt template for generating 10 questions for each triple

```
You are an intelligent system with access to a vast amount of
information. I will provide you with a knowledge graph in the
form of triples (subject, predicate, object).

Your task is to generate ten questions based on the kg.
The questions should assess understanding and insight into the
information presented in the graph.

Provide the output in JSON format, with each question having a unique
identifier. Instructions:
 1.Analyze the provided knowledge graph.
 2.Generate ten questions that are relevant to the information in kg.
 3.Provide the questions in JSON format, each with a unique identifier.

Input Knowledge Graph: Albert Einstein bornIn Ulm, Germany
```

```
Expected Response: {
  "questions": [
    {"id": 1,
    "question": "Where was Albert Einstein born?"},
    {"id": 2,
    "question": "What is Albert Einstein known for?"},
    {"id": 3,
    "question": "In what year was the Theory of Relativity published?"},
    {"id": 4,
    "question": "Where did Albert Einstein work?"},
    {"id": 5,
    "question": "What prestigious award did Albert Einstein win?"},
    {"id": 6,
    "question": "Which theory is associated with Albert Einstein?"},
    {"id": 7,
    "question": "Which university did Albert Einstein work at?"},
    {"id": 8,
    "question": "What did Albert Einstein receive the Nobel Prize in?"},
    {"id": 9,
    "question": "In what field did Albert Einstein win a Nobel Prize?"},
    {"id": 10,
    "question": "Name the city where Albert Einstein was born."}
]}
Considering the above information, please respond to this kg: {query}
The output should be in JSON format with each question having a unique
identifier and question doesn't contain term knowledge graph, without
any additional information
```

A.3 RAG Prompt

Prompt template for RAG

```
Context information is below.
---------------------
{context_str}
---------------------
Given the context information and without prior knowledge,
Evaluate whether the information in the documents supports the triple.
Please provide your answer in the form of a structured JSON
```

```
format containing a key "output" with the value as "yes" or "no".
If the triple is correct according to the documents, the value
should be "yes". If the triple is incorrect, the value should be "no".

{few_shot_examples}

Query: {query_str}
Answer:
```

A.4 Reasoning Prompt

Prompt template for generate reason

```
Context information is below.
---------------------
{context_str}
---------------------
Given the context information without your knowledge,
you evaluate that the fact '{fact}' is {correct/wrong}, now explain
your reasoning for your evaluation. write you're answer in this format

'The claim is {correct/wrong} because: 'REASON''

without any further description or writing fact again, consider that you
can't change you're mind and you should reason why it's {correct/wrong}
```

B

Chunking Strategies

As discussed in Section 5.4, the method used to chunk input text is a critical decision in the design of a RAG system. Here, we present concrete examples of how different chunking strategies affect the segmentation of text, using the *correct_award_00000* entry from the FactBench dataset. The text is as follows:

> *Henry Dunant award Nobel Peace Prize*

The model used for these examples is *Gemma2* with *similarity_top_k* set to 3, and *BAAI/bge-small-en-v1.5* as embedding model. The documents are selected using *ms-marco-MiniLM-L-6-v2* discussed in 5.2.2. We report the best node found by through our pipeline for each chunking strategy.

Table B.1: Evaluation of text segmentation using a chunk Size of 512, text chunks derived from the entry "Henry Dunant award Nobel Peace Prize".

Chunk	Score
Abstract When Jean Henry Dunant received the first Nobel Peace Prize in 1900, he was praised for "the supreme humanitarian achievement of the nineteenth century." This praise was merited, for Dunant had led the creation of both the international Red Cross and the First Geneva Convention. The Red Cross has since saved countless lives and relieved human suffering around the world. The Geneva Convention established that those treating war wounded, wearing a red cross, would not be attacked. With this Convention, Dunant began the creation of international humanitarian law to reduce the suffering caused by war. Despite Dunant's vital contributions, he has been largely forgotten. This article briefly tells the story of this dedicated humanitarian leader and of his great achievements. Recommended Citation McFarland, Sam (2017) "A Brief History of An Unsung Hero and Leader – Jean Henry Dunant and the Founding of the Red Cross at the Geneva Convention," International Journal of Leadership and Change: Vol. 5: Iss. 1, Article 5. Available at: https://digitalcommons.wku.edu/ijlc/vol5/iss1/5	90.5791
In 1901, Henry Dunant was co-awarded the first Nobel Peace Prize in recognition of his devotion to the humanitarian cause. (Español): Henry Dunant, Premio Nobel de la Paz - En 1901, Henry Dunant fue co-galardonado con el primer Premio Nobel de la Paz en reconocimiento a su devisión por la causa humanitaria. Credit: ICRC / Vincent Varin / www.icrc.org	90.5830
To celebrate the memory and work of Henry Dunant, on the centenary of the presentation of the first Nobel Peace Prize, rightly awarded to Dunant for his having founded the institution of the International Red Cross, this paper presents the reader with some insights into his activities and sufferings, his trials and tribulations, and the hope and strength of his character. The ceaseless efforts made by Dunant to bring about the Institution which today represents Hope for so many suffering people who are silent victims of wars and atrocities, are fleetingly presented. The authors' intention is to give due recognition to Dunant for his work, and to highlight the humanity and the moral and social worth of the face behind the International Red Cross.	90.6017

Table B.2: Evaluation of text segmentation using a Small to Big technique (base chunk size 1024), text chunks derived from the entry "Henry Dunant award Nobel Peace Prize".

Chunk	Score
Henry Dunant The Nobel Peace Prize 1901 Nobel co-recipient: Frédéric Passy Role: Founder of the International Committee of the Red Cross, Geneva, Originator Geneva Convention (Convention de Genève) Nobel Prize Cash and Philanthropy Jean Henry Dunant, though poor, donated his Nobel Prize money to charity. Hans Daae, a military physician, managed to get the money deposited in a bank in Norway. Thus Dunant's creditors could not claim the money. When Dunant was alive the money remained untouched in the bank. He lived frugally in a Swiss nursing home. Dunant's will bequeathed one half of the money to the Norwegian Red Cross and the Norwegian Women's Public Health Association. The will bequeathed the other half of the money to charities in Switzerland. Daae was also responsible for Dunant being awarded the Noble Prize.	90.6243

Table B.3: Evaluation of text segmentation using a Sliding Window with window size 3, text chunks derived from the entry "Henry Dunant award Nobel Peace Prize".

Window - Highlighted Text is Original Text
You can read more about that here: From the first Nobel Prize award ceremony, 1901 The announcement that the founder of the Red Cross had been chosen as Peace Prize laureate met with mixed reactions. Dunant had been awarded the prize for ameliorating the suffering of wounded soldiers, not for organising peace congresses or reducing standing forces, as stipulated in Alfred Nobel's will. The Nobel Committee had chosen a broad interpretation of the provision that a laureate should "further fraternity between nations". The Red Cross: three-time recipient of the Peace Prize Henry Dunant (1828–1910). Switzerland, "for his humanitarian efforts to help wounded soldiers and create international understanding" Frédéric Passy (1822–1912). France, "for his lifelong work for international peace conferences, diplomacy and arbitration."
On 10th of December 1901 the first Nobel Peace Prize was awarded. It went to Henry Dunant, founder of the International Committee of the Red Cross, who shared the first Nobel Peace Prize with Frédéric Passy, a leading international pacifist of the time. Since then, the Red Cross has been awarded the Peace Prize three times. The Red Cross: Three-time recipient of the Peace Prize Four of them given out in Stockholm and one, the Peace Prize, in Christiania, as Oslo was then called. You can read more about that here: From the first Nobel Prize award ceremony, 1901 The announcement that the founder of the Red Cross had been chosen as Peace Prize laureate met with mixed reactions. Dunant had been awarded the prize for ameliorating the suffering of wounded soldiers, not for organising peace congresses or reducing standing forces, as stipulated in Alfred Nobel's will.
Henry Dunant The Nobel Peace Prize 1901 Nobel co-recipient: Frédéric Passy Role: Founder of the International Committee of the Red Cross, Geneva, Originator Geneva Convention (Convention de Genève) Nobel Prize Cash and Philanthropy Jean Henry Dunant, though poor, donated his Nobel Prize money to charity. Hans Daae, a military physician, managed to get the money deposited in a bank in Norway. Thus Dunant's creditors could not claim the money. When Dunant was alive the money remained untouched in the bank. He lived frugally in a Swiss nursing home. Dunant's will bequeathed one half of the money to the Norwegian Red Cross and the Norwegian Women's Public Health Association.

References

[1] Meta AI. *Introducing LLaMA 3: Advancing Open Foundation Models*. https://ai.meta.com/blog/meta-llama-3-1/. Accessed: 2024-10-17. 2023.

[2] Hyung Won Chung et al. *Scaling Instruction-Finetuned Language Models*. 2022. DOI: 10.48550/ARXIV.2210.11416. URL: https://arxiv.org/abs/2210.11416.

[3] Alexis Conneau et al. "Unsupervised Cross-lingual Representation Learning at Scale". In: *CoRR* abs/1911.02116 (2019). arXiv: 1911.02116. URL: http://arxiv.org/abs/1911.02116.

[4] 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.

[5] Alphaeus Dmonte et al. *Claim Verification in the Age of Large Language Models: A Survey*. 2024. arXiv: 2408.14317 [cs.CL]. URL: https://arxiv.org/abs/2408.14317.

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

[7] Luis Galárraga et al. "AMIE: Association rule mining under incomplete evidence in ontological knowledge bases". In: May 2013, pp. 413–422. DOI: 10.1145/2488388.2488425.

[8] Daniel Gerber et al. "DeFacto—Temporal and multilingual Deep Fact Validation". In: *Journal of Web Semantics* 35 (2015). Machine Learning and Data Mining for the Semantic Web (MLDMSW), pp. 85–101. ISSN: 1570-8268. DOI: https://doi.org/10.1016/j.websem.2015.08.001. URL: https://www.sciencedirect.com/science/article/pii/S1570826815000645.

[9] Michael Günther et al. *Jina Embeddings 2: 8192-Token General-Purpose Text Embeddings for Long Documents*. 2024. arXiv: 2310.19923 [cs.CL]. URL: https://arxiv.org/abs/2310.19923.

[10] Edward J Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

[11] Gautier Izacard et al. *Unsupervised Dense Information Retrieval with Contrastive Learning*. 2022. arXiv: 2112.09118 [cs.IR]. URL: https://arxiv.org/abs/2112.09118.

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

[13] M. Abdul Khaliq et al. *RAGAR, Your Falsehood Radar: RAG-Augmented Reasoning for Political Fact-Checking using Multimodal Large Language Models*. 2024. arXiv: 2404.12065 [cs.CL]. URL: https://arxiv.org/abs/2404.12065.

[14] Aditya Kusupati et al. *Matryoshka Representation Learning*. 2024. arXiv: 2205.13147 [cs.LG]. URL: https://arxiv.org/abs/2205.13147.

[15] Nayeon Lee et al. *Factuality Enhanced Language Models for Open-Ended Text Generation*. 2023. arXiv: 2206.04624 [cs.CL]. URL: https://arxiv.org/abs/2206.04624.

[16] Zehan Li et al. *Towards General Text Embeddings with Multi-stage Contrastive Learning*. 2023. arXiv: 2308.03281 [cs.CL]. URL: https://arxiv.org/abs/2308.03281.

[17] Jerry Liu. *Tweet on Information Retrieval and LLMs*. Accessed: 2024-10-10. 2023. URL: https://twitter.com/jerryjliu0/status/1708147687084986504.

[18] Shayne Longpre, Yi Lu, and Joachim Daiber. *MKQA: A Linguistically Diverse Benchmark for Multilingual Open Domain Question Answering*. 2020. URL: https://arxiv.org/pdf/2007.15207.pdf.

[19] Stefano Marchesin, Gianmaria Silvello, and Omar Alonso. "Utility-Oriented Knowledge Graph Accuracy Estimation with Limited Annotations: A Case Study on DBpedia". In: *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* 12.1 (Oct. 2024), pp. 105–114. DOI: 10.1609/hcomp.v12i1.31605. URL: https://ojs.aaai.org/index.php/HCOMP/article/view/31605.

[20] Mistral AI Team. *Announcing Mistral 7B*. Accessed: 2024-10-17. 2023. URL: https://mistral.ai/news/announcing-mistral-7b/.

[21] John X. Morris and Alexander M. Rush. *Contextual Document Embeddings*. 2024. arXiv: 2410.02525 [cs.CL]. URL: https://arxiv.org/abs/2410.02525.

[22] Niklas Muennighoff et al. "MTEB: Massive Text Embedding Benchmark". In: *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Ed. by Andreas Vlachos and Isabelle Augenstein. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 2014–2037. DOI: 10.18653/v1/2023.eacl-main.148. URL: https://aclanthology.org/2023.eacl-main.148.

[23] Prakhar Ojha and Partha Talukdar. "KGEval: Accuracy Estimation of Automatically Constructed Knowledge Graphs". 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. 1741–1750. DOI: 10.18653/v1/D17-1183. URL: https://aclanthology.org/D17-1183.

[24] 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: https://arxiv.org/abs/1908.10084.

[25] Stephen Robertson et al. "Okapi at TREC-3." In: Jan. 1994, pp. 0-.

[26] Jon Saad-Falcon et al. *Benchmarking and Building Long-Context Retrieval Models with LoCo and M2-BERT*. 2024. arXiv: 2402.07440 [cs.IR]. URL: https://arxiv.org/abs/2402.07440.

[27] Soumya Sanyal et al. *Are Machines Better at Complex Reasoning? Unveiling Human-Machine Inference Gaps in Entailment Verification*. 2024. arXiv: 2402.03686 [cs.CL]. URL: https://arxiv.org/abs/2402.03686.

[28] Saba Sturua et al. *jina-embeddings-v3: Multilingual Embeddings With Task LoRA*. 2024. arXiv: 2409.10173 [cs.CL]. URL: https://arxiv.org/abs/2409.10173.

[29] Jianlin Su et al. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. 2023. arXiv: 2104.09864 [cs.CL]. URL: https://arxiv.org/abs/2104.09864.

[30] Fabian Suchanek et al. *YAGO 4.5: A Large and Clean Knowledge Base with a Rich Taxonomy*. 2024. arXiv: 2308.11884 [cs.AI]. URL: https://arxiv.org/abs/2308.11884.

[31] Gemma Team et al. *Gemma 2: Improving Open Language Models at a Practical Size*. 2024. arXiv: 2408.00118 [cs.CL]. URL: https://arxiv.org/abs/2408.00118.

[32] Qwen Team. *Qwen2.5: A Party of Foundation Models*. Sept. 2024. URL: https://qwenlm.github.io/blog/qwen2.5/.

[33] Nandan Thakur et al. *BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models*. 2021. arXiv: 2104.08663 [cs.IR]. URL: https://arxiv.org/abs/2104.08663.

[34] Liang Wang et al. *Improving Text Embeddings with Large Language Models*. 2024. arXiv: 2401.00368 [cs.CL]. URL: https://arxiv.org/abs/2401.00368.

[35] Liang Wang et al. *Multilingual E5 Text Embeddings: A Technical Report*. 2024. arXiv: 2402.05672 [cs.CL]. URL: https://arxiv.org/abs/2402.05672.

[36] Shitao Xiao et al. *C-Pack: Packaged Resources To Advance General Chinese Embedding*. 2023. arXiv: 2309.07597 [cs.CL].

[37] An Yang et al. "Qwen2 Technical Report". In: *arXiv preprint arXiv:2407.10671* (2024).

[38] Zhenrui Yue et al. *Retrieval Augmented Fact Verification by Synthesizing Contrastive Arguments*. 2024. arXiv: 2406.09815 [cs.CL]. URL: https://arxiv.org/abs/2406.09815.

[39] Xin Zhang et al. *mGTE: Generalized Long-Context Text Representation and Reranking Models for Multilingual Text Retrieval*. 2024. arXiv: 2407.19669 [cs.CL]. URL: https://arxiv.org/abs/2407.19669.

[40] Xinyu Zhang et al. "MIRACL: A Multilingual Retrieval Dataset Covering 18 Diverse Languages". In: *Transactions of the Association for Computational Linguistics* 11 (2023), pp. 1114–1131. DOI: 10.1162/tacl_a_00595. URL: https://aclanthology.org/2023.tacl-1.63.

[41] Xuan Zhang and Wei Gao. *Towards LLM-based Fact Verification on News Claims with a Hierarchical Step-by-Step Prompting Method*. 2023. arXiv: 2310.00305 [cs.CL]. URL: https://arxiv.org/abs/2310.00305.

Acknowledgments