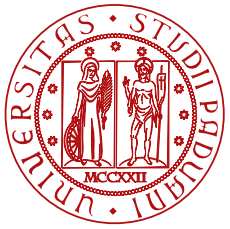

Università degli Studi di Padova

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 accuracy of data inside knowledge graphs is essential in a time when information management is depending more and more on automated methods. This thesis uses retrieval-augmented generating (RAG) to provide a fresh method for fact verification in knowledge graphs. Combining large language models, information retrieval methods, and a multi-stage verification procedure, the proposed pipeline evaluates the veracity of facts expressed in knowledge graphs. Important elements consist of query generation from knowledge graph triples, web search integration for external evidence retrieval, embedding-based document chunking and retrieval, and an ensemble of language models for fact verification. The system synthesizes outputs from several models by using adaptive dispute resolution techniques and majority voting. Extensive studies assess several text chunking techniques, embedding models, and document choosing procedures. Results show areas for more optimization as well as the efficiency of the suggested method in verifying knowledge graph truths. This effort advances the developing domains 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

1 Introduction

In the age of big data and artificial intelligence, the capacity to effectively verify facts and evaluate the veracity of information has become increasingly essential. Knowledge graphs, which depict knowledge through entities and their interrelations, have arisen as a formidable instrument for structuring and querying enormous volumes of structured data. Nonetheless, guaranteeing the precision and dependability of the information within these knowledge graphs continues to pose a considerable difficulty. This thesis introduces an innovative method for verifying facts in knowledge graphs through retrieval-augmented generation, integrating the advantages of Information Retrieval (IR), natural language processing, and machine learning.

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 Machine Learning (ML) and Natural Language Processing (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. [12] 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 [1] 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 [12].

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. [2] provide a comprehensive survey ofcLLM-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.

Several studies have demonstrated the effectiveness of LLM-based approaches in claim verification:

- Zhang and Gao [17] introduced HiSS, a hierarchical prompting technique that improved performance on complex news claim verification tasks.
- Lee et al. [8] 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

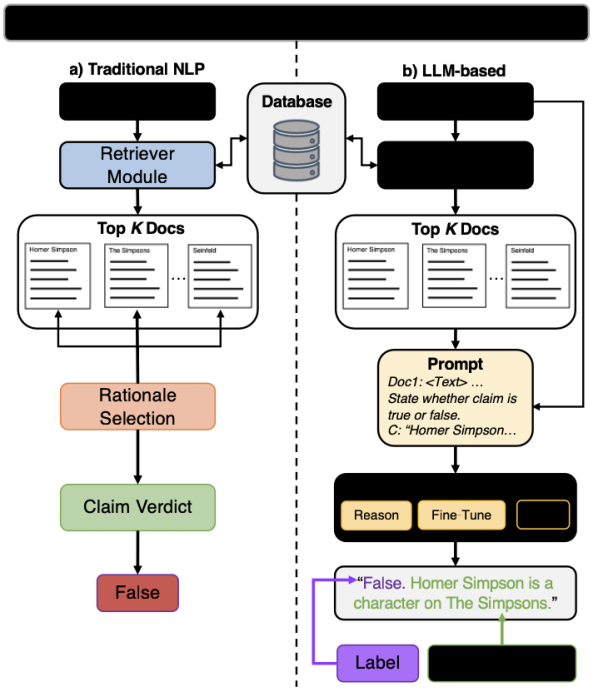

Figure 2.2: Comparison of claim verification systems between NLP-based (traditional) and LLM-based for claim veracity. [2].

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. Explainability Focus: While many LLM approaches provide explanations, our method places a stronger emphasis on generating detailed, step-by-step justifications for veracity assessments.
4. Specialized Embedding Models: We utilize domain-specific embedding models for improved semantic understanding in the retrieval phase, which is not commonly discussed in general LLM-based approaches.

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 [16] 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

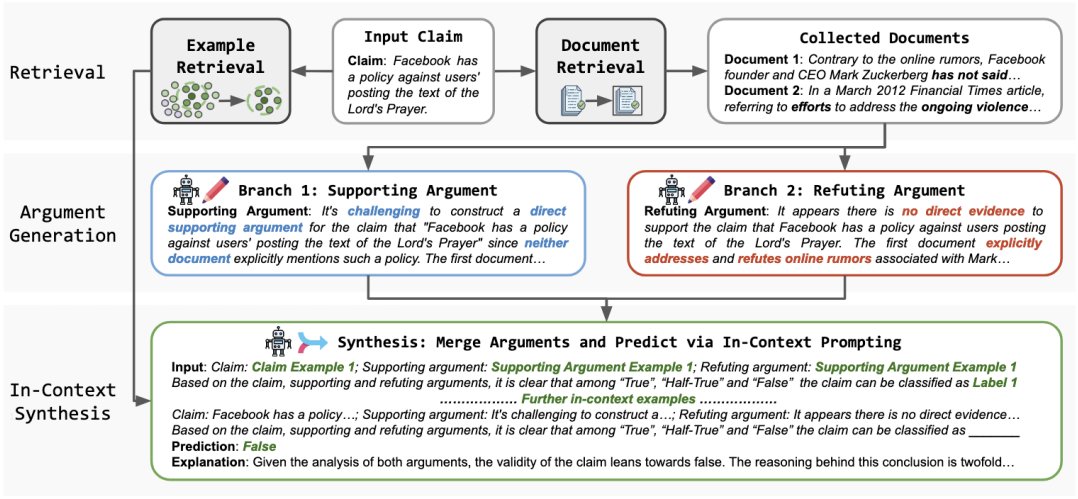

Figure 2.3: The proposed RAFTS [16], which performs few-shot fact verification by incorporating informative in-context demonstrations and contrastive arguments with nuanced information derived from the retrieved documents

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 [7] 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.

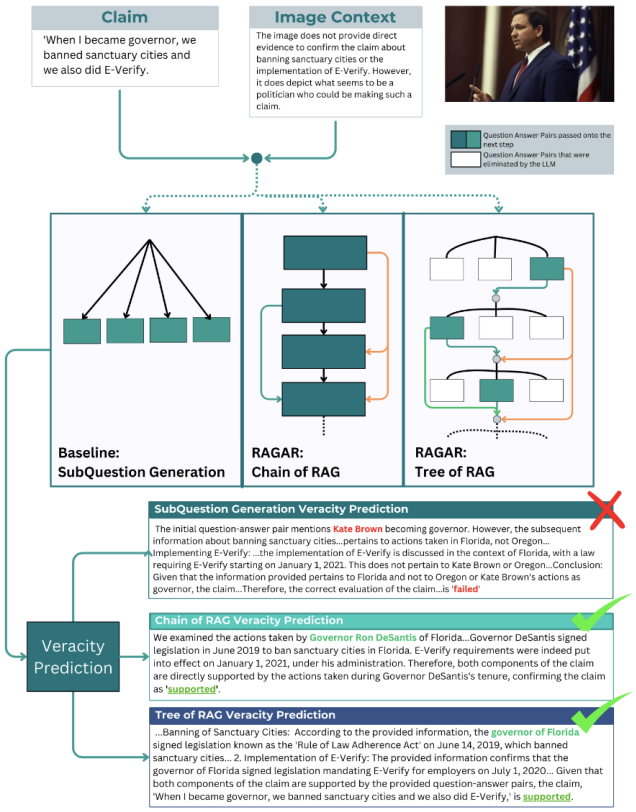

Figure 2.4: An overview of the fact-checking pipeline [7] contrasting the baseline Sub-Question Generation approach from the RAGAR: Chain of RAG and RAGAR: Tree of RAG approach followed by a veracity explanation generated by a Veracity Prediction module.

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 ac-

curacy, 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 presents a detailed examination of a multi-stage NLP pipeline designed to enhance information retrieval and question answering capabilities. The pipeline integrates various cutting-edge technologies and methodologies, creating a synergistic system that pushes the boundaries of what is possible in automated information processing and response generation. However, the challenges of accurately interpreting user queries, retrieving relevant information from vast datasets, and generating coherent and contextually appropriate responses remain significant. This pipeline addresses these challenges through a carefully orchestrated series of processes, each designed to refine and enhance the quality of information flow from input to output.

At its core, the pipeline leverages a knowledge graph dataset, serving as the foundational repository of interconnected information. This graph structure allows for the representation of complex relationships between entities, facilitating more nuanced understanding and retrieval of information. The pipeline then employs a series of sophisticated mechanisms, including query generation, cross-encoding for relevance assessment, and multi-tiered information retrieval strategies, to navigate this knowledge landscape effectively.

One of the key innovations in this pipeline is its approach to context processing and information synthesis. By breaking down retrieved information into manageable chunks and employing advanced embedding techniques, the system can perform more granular and accurate analyses of textual data. This granularity, combined with the implementation of multiple LLMs working in concert, allows for a more robust and nuanced interpretation of complex queries

and generation of comprehensive responses.

The integration of external information sources, notably through the incorporation of Google Search capabilities, further enhances the pipeline's ability to access and process up-to-date and diverse information. This hybrid approach, combining structured knowledge graphs with dynamic web-based information retrieval, positions the pipeline at the forefront of adaptive and responsive AI systems.

Critical to the pipeline's effectiveness is its sophisticated decision-making architecture. Employing strategies such as majority voting among multiple models and the implementation of a final judge for conflict resolution, the system strives to achieve a balance between diverse perspectives and the need for coherent, unified outputs. This approach not only enhances the accuracy and reliability of the generated responses but also provides a framework for managing the inherent uncertainties and potential biases in AI-driven decision-making processes.

As we delve deeper into each component of this pipeline, it is crucial to maintain a critical perspective on both its capabilities and limitations. While the system represents a significant advancement in NLP and information retrieval technologies, it also raises important questions about the ethical implications of AI-driven information processing, the potential for bias in knowledge representation and model training, and the broader societal impacts of increasingly sophisticated question-answering systems.

This chapter aims to provide a comprehensive analysis of each stage of the pipeline, 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 (e.g., "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 (ie. LLama3 [3], Gemma2 [14]) 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 [4] dataset may be easily comprehended by concatenating the subject, predicate, and object of each triple.

Knowledge Graph			**Source**	**Is Generated ?**	**Final Text**
Subject	Predicate	Object			
Albert Einstein	*Birth Place*	*Ulm, Germany*	FactBench [4]	✗	*Albert Einstein birth place Ulm, Germany*
Chris Benoit	*deathPlace*	*Fayetteville, Georgia*	FactBench [4]	✗	*Chris Benoit death place Fayetteville, Georgia*
Alexander_III _of_Russia	*isMarriedTo*	*Maria_Feodorovna __Dagmar_of_Denmark_*	YAGO [13]	✓	*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.*

Table 3.1: Generation of Human-Understandable Text

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.

Implementation of this technique likely involves: Using **LLms!** (**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 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 3.1, this module takes multiple inputs and produces relevance scores for each question.

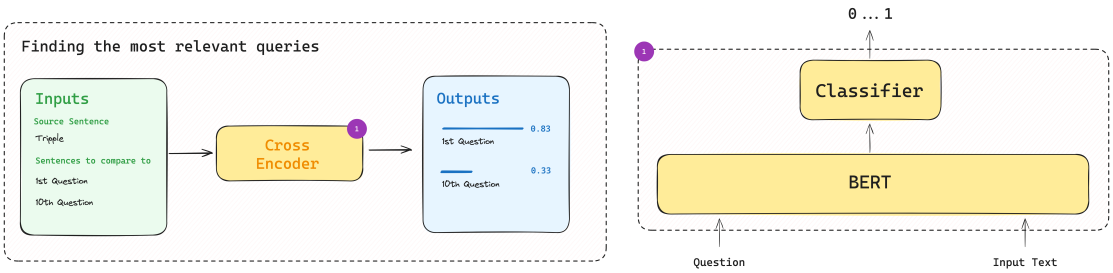

Figure 3.1: Cross-Encoder Architecture

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 3.1) 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 [5] 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.

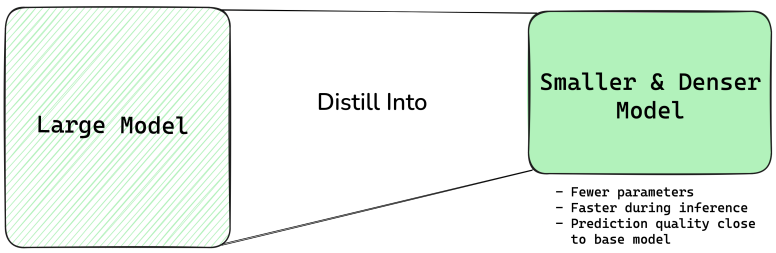

Figure 3.2: Knowledge Distillation Process

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

Table 3.2: Question Generation and Scoring Procedure

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.

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.

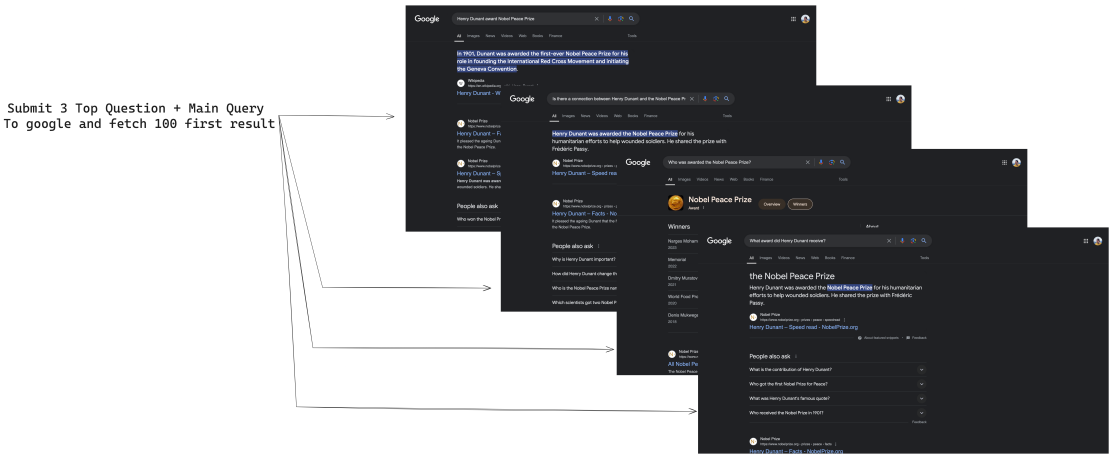

Figure 3.3: Fetching the results from Google Search

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.

```
1 import os
2 import grequests
```

[2]https://pypi.org/project/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 *newspaper4k*[5] library to extract the content of the page. *newspaper4k* is a Python

[3] https://www.selenium.dev/
[4] https://scrapy.org/
[5] https://newspaper4k.readthedocs.io/en/latest/

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.

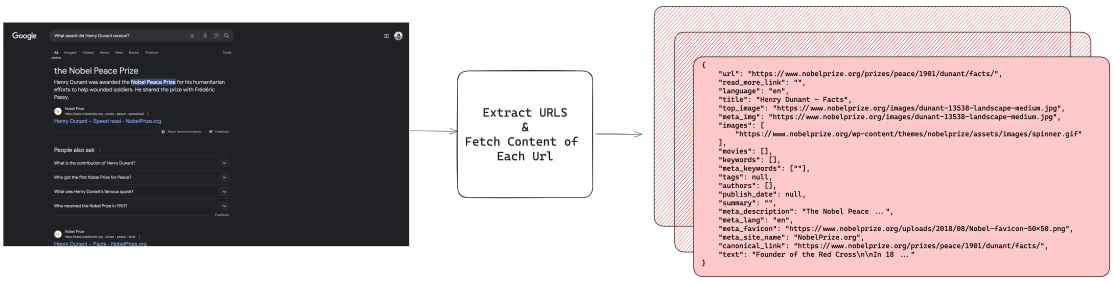

Figure 3.4: Extracted Content from the Crawled URLs using newspaper4k

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:

- **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. More details about the data filtering process methodology are provided in the section 5.2.

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 S̈maller Chunksẅuggests 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.

we use the *SentenceWindowNodeParser* to parse documents into single sentences per node. Each node also contains a *"window"* with the sentences on either side 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.

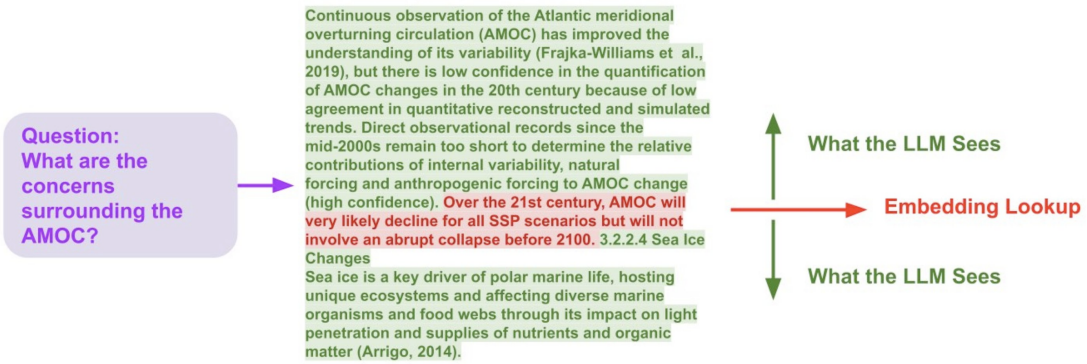

Figure 3.5: Node sentence window replacement [9]

There is example of the *SentenceWindowNodeParser* and *MetadataReplacementNodePostProcessor* in the Appendix B.3.

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 large language models (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:

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

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

3.5.3 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.

3.6 Model Diversity 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 Model Diversity

As illustrated in the pipeline diagram, the system incorporates a variety of models, including both paid models and large parameter models.

This diversity serves several critical functions: Complementary Strengths: Different model architectures and training paradigms often excel in distinct areas, allowing the system to leverage specialized capabilities for various tasks. Bias Mitigation: By incorporating diverse models, the system can potentially offset individual model biases, leading to more balanced outputs. Robustness: Model diversity enhances the system's ability to handle a wide range of queries and scenarios, improving overall performance and reliability. Innovation Integration: The inclusion of different model types allows for the rapid integration of state-of-the-art innovations in the field of NLP.

Key aspects of model diversity in the pipeline: a) Paid Models:

Potentially include proprietary or subscription-based models with specific optimizations or capabilities. May offer enhanced performance in certain domains or tasks. Could provide additional layers of quality assurance or specialized features.

b) Large Parameter Models:

Likely refer to models with billions of parameters, such as GPT-3, PaLM, or similar architectures. Offer broad knowledge coverage and sophisticated language understanding capabilities. Excel in tasks requiring generalization and handling of diverse contexts.

Implementation considerations:

Model Selection Criteria: Developing a framework for selecting and integrating models based on their complementary strengths and overall contribution to system performance. Resource Management: Balancing the computational demands of running multiple, potentially resource-intensive models. Versioning and Updates: Establishing protocols for managing model versions and incorporating updates or new models into the existing ecosystem.

3.7.2 Conflict Resolution Strategies The pipeline incorporates several mechanisms for resolving conflicts that may arise from divergent model outputs:

Majority Voting System: As previously discussed, 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: a) Conflict Identification:

Detecting discrepancies or contradictions in outputs from different models. Assessing the degree of disagreement and determining when intervention is necessary.

b) Resolution Mechanisms:

Employing heuristics or learned strategies to reconcile conflicting information. Potentially utilizing confidence scores or uncertainty estimates from individual models to inform decision-making.

c) Tie-Breaking:

Implementing strategies for scenarios where the majority voting system results in a tie or lacks a clear consensus.

d) Consistency Enforcement:

Ensuring that the final output maintains logical consistency and aligns with established knowledge bases.

e) Explanation Generation:

Potentially providing rationales for conflict resolution decisions, enhancing the explainability of the system.

3.7.3 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." 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.

3.7.4 Ethical Considerations and Challenges The implementation of diverse models and conflict resolution mechanisms raises several ethical considerations and challenges:

Transparency: Ensuring that the conflict resolution process, especially when involving proprietary models, maintains a degree of transparency and explainability. Bias Amplification: Monitoring and mitigating potential scenarios where the conflict resolution process might inadvertently amplify certain biases present in multiple models. Accountability: Establishing clear lines of accountability for decisions made by the final judge, especially in sensitive or high-stakes scenarios. Data Privacy: Ensuring that the conflict resolution process, particularly when escalating to more advanced models, adheres to data privacy regulations and ethical guidelines. Model Disagreement Handling: Developing strategies for scenarios where persistent disagreements between models might indicate underlying issues in the knowledge base or query interpretation. Continuous Evaluation: Implementing mechanisms for ongoing assessment of the conflict resolution strategies to ensure they remain effective and unbiased over time.

In conclusion, the Model Diversity and Conflict Resolution components of the pipeline represent a sophisticated approach to leveraging the strengths of various AI models while mitigating their individual weaknesses. By employing a combination of majority voting, a dedicated final judge, and adaptive escalation to more advanced models, the system aims to produce reliable, consistent, and high-quality outputs. This approach not only enhances the robustness of the question-answering system but also paves the way for the integration of increasingly advanced AI models in a controlled and effective manner. However, it also necessitates ongoing attention to ethical considerations and the development of transparent, accountable processes for managing the complexities introduced by model diversity and conflict resolution.

For our pipeline we use the *Majority Voting System* to select the final response, the system selects the response with the highest score from the LLMs, and if there is a tie, the system selects the response with two different approaches:

- **Adaptive models:** The system uses algorithm 1 to select model based on

the user specified settings.

- **Commercial models:** The system selects the response from the commercial models, as they are more accurate and reliable (e.g., GPT-4).

3.7 Pipeline Flow and Decision Points

The architecture of our natural language processing 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, 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:

- **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.

Algorithm 1 Resolve Ties in Majority Voting System

Require:
$data$ - A list of dictionaries, each representing a model's response
$finalJudger$ - A dictionary mapping file indices to final model
$atLeast$ - A boolean flag:
True: select file(s) with highest agreement score
False: select file(s) with lowest agreement score

procedure ProcessFiles($data, finalJudger, atLeast$)
  $majorityValues \leftarrow \emptyset$
  $keys \leftarrow$ keys from first element of $data$
  for each key in $keys$ **do**
    $values \leftarrow$ list of values for key from all data
    $count \leftarrow$ count occurrences of 1's and 0's in $values$
    if count of 1's > count of 0's **then**
      $majorityValues[key] \leftarrow 1$
    else if count of 1's < count of 0's **then**
      $majorityValues[key] \leftarrow 0$
    else
      continue to next key
    end if
  end for
  $modelScores \leftarrow \emptyset$
  for $i \leftarrow 0$ to $|data| - 1$ **do**
    $trueCount \leftarrow 0$
    for each (k, v) in $data[i]$ **do**
      if $majorityValues[k] = v$ **then**
        $trueCount \leftarrow trueCount + 1$
      end if
    end for
    Add $(i, trueCount)$ to $modelScores$
  end for
  if $atLeast$ is True **then**
    $maxScore \leftarrow$ maximum score in $modelScores$
    $candidates \leftarrow$ indices with score equal to $maxScore$
  else
    $minScore \leftarrow$ minimum score in $modelScores$
    $candidates \leftarrow$ indices with score equal to $minScore$
  end if
  $chosenIndex \leftarrow$ random choice from $candidates$
  return $finalJudger[chosenIndex]$
end procedure

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	
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.
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.

Table 3.3: Comprenhensive list of decision points in the pipeline flow.

In conclusion, the Pipeline Flow and Decision Points represent a complex yet well-structured approach to natural language processing and question answer-

ing. 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.

3.8 Performance Metrics and Evaluation

A crucial aspect of any advanced natural language processing pipeline is the rigorous evaluation of its performance. This section explores the various metrics, methodologies, and considerations involved in assessing the effectiveness and reliability of our question-answering system.

3.8.1 PASSED and FAILED Criteria

As depicted in the pipeline diagram, the system incorporates explicit PASSED and FAILED 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.

Key aspects of this evaluation approach:

Definition of Success Criteria: Establishing clear, quantifiable benchmarks for what constitutes a "PASSED" state. Potentially incorporating multiple factors such as relevance, accuracy, and coherence of responses. Failure Analysis: Detailed examination of cases resulting in a "FAILED" state to identify patterns or systemic issues. Utilization of failure instances for continuous improvement and refinement of the pipeline. Threshold Setting: Determining appropriate thresholds for PASSED/FAILED states, balancing between stringency and practical usability. Potential implementation of graduated levels of success/failure for more nuanced evaluation.

3.8.2 Relevance and Accuracy Metrics

While not explicitly shown in the image, the evaluation of a question-answering system typically involves assessing both the relevance and accuracy of responses.

Potential metrics include:

Precision: The proportion of relevant and correct information in the response. Recall: The proportion of relevant information from the source material included in the response. F1 Score: The harmonic mean of precision and recall, providing a balanced measure of accuracy. Mean Reciprocal Rank (MRR): For systems providing ranked answers, MRR assesses how high the correct answer appears in the ranking.

Implementation considerations:

Automated Scoring: Developing algorithms for automated assessment of relevance and accuracy. Human Evaluation: Incorporating expert human judgment for a subset of responses to validate automated metrics.

3.8.3 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). Resource Utilization: Monitoring computational resource usage, particularly important given the use of multiple LLMs.

3.8.4 Consistency and Coherence Evaluation

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

Inter-model Agreement Rate: Measuring the frequency of consensus among different LLMs. Coherence Scoring: Assessing the logical flow and internal consistency of generated responses. Stability Across Queries: Evaluating the consistency of responses to similar or related queries over time.

3.8.5 Robustness and Edge Case Handling

Assessing the pipeline's performance under various conditions:

Stress Testing: Evaluating performance under high query volumes or with particularly complex inputs. Adversarial Inputs: Testing the system's resilience to intentionally challenging or malformed queries. Domain Adaptation: Assessing how well the system performs across different subject areas or specialized domains.

4 Analysis

4.1 A section

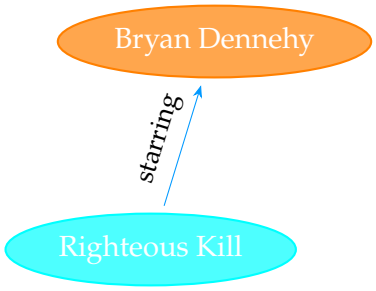

Figure 4.1: Image created with TikZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

```
1 import numpy as np
2
3 def incmatrix(genl1,genl2):
4     m = len(genl1)
5     n = len(genl2)
6     M = None #to become the incidence matrix
```

```
    VT = np.zeros((n*m,1), int)  #dummy variable

    test = "String"

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

Code 4.1: Code snippet example

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 proposed in recent literature. We will focus on three key components of our framework: 1) the chunking strategies used to segment information, 2) the embedding models employed for representation, and 3) the retrieval mechanisms utilized to fetch relevant information. 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, while also considering the efficacy of sampling methods compared to full data runs.

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. This includes, but is not limited to, testing different sampling methods against full data runs.
3. **Performance Evaluation:** For each configuration, we assess the system's performance using predefined metrics (detailed in Section 5.1.3).
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 Sampling Methods Evaluation

As one of the parameters under investigation, we compare various sampling methods to full data runs:

- **Full Data Runs:** Establish a baseline using the entire dataset.
- **Simple Random Sampling:** Randomly select a subset of n data points from a population of size N.

- **Unique Over Sampling:** Evenly distributes a specified number of samples across different categories in a dataset. It ensures minimal duplication by prioritizing unique samples and filling the remainder slots using random sampling when necessary.

This comparison aims to determine if sampling can reduce computational costs and accelerate results without significant loss in accuracy.

5.1.3 Evaluation Metrics

The performance of each configuration is assessed using the following metrics:

- **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 Score (Avg):** Calculated based on accuracy across multiple runs:

$$\text{Avg} = \frac{1}{m} \sum_{i=1}^{m} \text{Acc}_i \tag{5.3}$$

  where m is the number of runs.

- **Average Latency:** Measured in seconds per query to assess computational efficiency:

$$\text{Avg Latency} = \frac{\text{Total Processing Time}}{\text{Number of Queries}} \tag{5.4}$$

5.1.4 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 D

ocument 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.

5.2.1 Unsupervised Methods

BM25

BM25 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.

Contriever

Contriever is a more recently proposed unsupervised method by Izacard et al.[6] 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,

Algorithm 2 BM25-based Sentence Retrieval

```
procedure PREPROCESS(sentence)
    Convert sentence to lowercase
    Remove punctuation
    Tokenize sentence into words
    Remove stopwords
    return preprocessed tokens
end procedure
procedure FETCHSIMILARSENTENCES(sentences, top_n)
    for each sentence in sentences do
        preprocessed ← Preprocess(sentence)
        Add preprocessed to tokenized_sentences
    end for
    bm25 ← CreateBM25Object(tokenized_sentences)
    query ← tokenized_sentences[0]
    scores ← bm25.GetScores(query)
    Sort scored_sentences by score in descending order
    result ← Top top_n sentences from result
end procedure
```

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

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

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 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 [10] spanning 26 languages, while also exhibiting superior NDCG@10 scores on English-language tasks in the BEIR benchmark [15]. 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 Retrieval-Augmented Generation (RAG) systems, and even in programming and software development support.

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

MS MARCO MiniLM

The MS MARCO MiniLM is another supervised neural model, based on the popular BERT architecture but distilled to a smaller size for efficiency.

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

Table 5.1: Performance of Pre-trained Cross-Encoders

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 [11], 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

With checking the similarity between the retrieved documents and the query in different methods, based on figure **??** we can figured out that Bm25-Okapi stands out as the most distinct model, with low similarity scores (0.16-0.19) to the others, suggesting it employs fundamentally different retrieval mechanisms. In contrast, the neural models show higher inter-model similarities (0.29-0.43), indicating shared approaches or architectures. The strongest relationship (0.43) is between contriever-msmarco and re-ranker-msmarco, likely due to shared training data or similar optimizations. The two re-ranker models also show high similarity (0.41), suggesting comparable re-ranking strategies. In general,

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

we can find out with different methods we have different result over the same query.

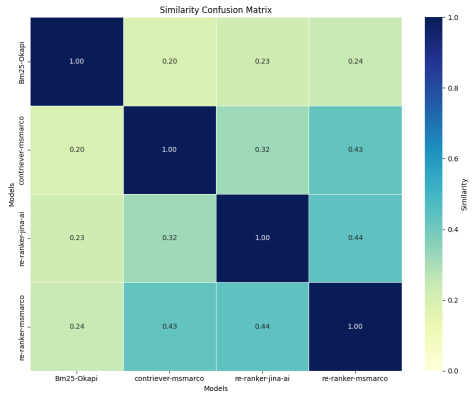

Figure 5.1: Document Retrieval Confusion Matrix

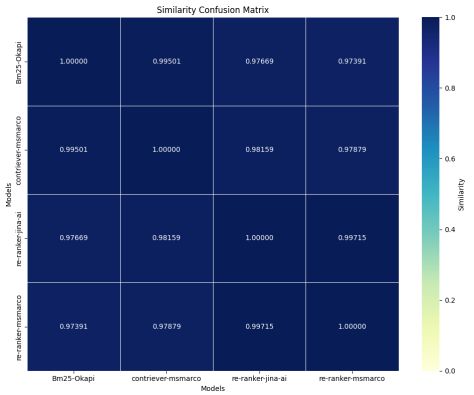

Figure 5.2: Document Retrieval Performance

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.

Retrieval Method, BAAI/bge-small-en-v1.5, Sliding Window - 6 , With Similarity Cut-off

Method	**Random Sampling**		**Over Sampling**	**Avg.**		**Complete Run**		
	Acc	F1	Acc	Acc	F1	Acc	F1	Latency
Unsupervised								
Bm25	0.719	0.505	**0.450**	**0.212**	0.255	0.8882	0.8940	0.353
contriever-msmarco	0.505	**0.450**	**0.212**	**0.528**	0.255	0.8932	0.8988	0.353
supervised								
jina-reranker-v2-base-multilingual	0.719	**0.450**	**0.212**	0.255	0.255	0.9004	0.9065	0.9065
ms-marco-MiniLM-L-6-v2	0.719	0.505	**0.450**	**0.212**	0.255	**0.9014**	**0.9077**	0.353

Table 5.2: Evaluation Results for Different Methods through the Pipeline (just with the Gemma2 model)

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 F1 score relative to the other models under consideration.

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 natural language processing (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 Alibaba-NLP/gte-large-en-v1.5

Model Overview

The gte-large-en-v1.5 model is part of the gte-vl.5 series released by the Institute for Intelligent Computing at Alibaba Group [Zhang et al., 2023]. It is

built upon a transformer++ encoder backbone, which enhances the standard BERT architecture [Devlin et al., 2018] with rotary position embeddings (RoPE) [Su et al., 2021] and gated linear units (GLU) [Dauphin et al., 2017]. The model supports input sequences up to 8192 tokens, a significant improvement over previous multilingual encoders limited to 512 tokens.

Training Methodology

gte-large-en-v1.5 undergoes a three-stage training pipeline:

- Masked language modeling (MLM) pre-training on the C4 dataset
- Weakly-supervised contrastive pre-training on GTE pre-training data
- Supervised contrastive fine-tuning on GTE fine-tuning data

The MLM pre-training follows a two-stage curriculum to efficiently handle long sequences. It first trains on shorter 512 token inputs, then resamples the data to include more long sequences and continues MLM on 8192 token inputs. The contrastive pre-training and fine-tuning stages utilize diverse text pair datasets to learn general-purpose text representations.

Evaluation

gte-large-en-v1.5 demonstrates strong performance on the MTEB multilingual benchmark, achieving state-of-the-art results in its model size category. It also shows competitive results on the LoCo long-context retrieval benchmark. The model strikes a good balance between embedding quality and computational efficiency.

5.3.2 jinaai/jina-embeddings-v3

Model Overview

jina-embeddings-v3 is a multilingual multi-task text embedding model developed by Jina AI [Sturm et al., 2023]. Based on the XLM-RoBERTa architecture [Conneau et al., 2019], it incorporates rotary position embeddings (RoPE) to handle input sequences up to 8192 tokens. A key feature is the inclusion of 5 LoRA (low-rank adaptation) [Hu et al., 2021] adapters to efficiently generate task-specific embeddings for retrieval, clustering, classification, and text matching.

Training Methodology

jina-embeddings-v3 is first pre-trained using masked language modeling on a large multilingual corpus. It then undergoes contrastive pre-training on weakly-supervised text pairs mined from web data. inally, the model is fine-tuned using supervised contrastive learning on annotated datasets for specific tasks. The training leverages recent techniques like Matryoshka embeddings [Kusupati et al., 2022] which allow flexibly truncating the embedding size to reduce storage costs.

Evaluation

On the MTEB benchmark, jina-embeddings-v3 outperforms other multilingual models like XLMR and achieves state-of-the-art results on several English tasks, surpassing large models from OpenAI and Cohere. Its strong cross-lingual transfer and flexible embedding size options make it highly practical for real-world applications.

5.3.3 dunzhang/stella_en_1.5B_v5

Model Overview

stella_en_1.5B_v5 is an English-specific embedding model built by Baai based on Alibaba's gte-large-en-vl.5 [Dunzhang, 2023]. It simplifies the usage of prompts, providing two general templates (one for sentence-to-passage and one for sentence-to-sentence tasks). Through Matryoshka representation learning, the model supports elastic embeddings with dimensions ranging from 32 to 8192, allowing users to easily trade off between embedding size and quality.

Training Methodology

stella_en_1.5B_v5 follows a similar training pipeline as gte-large-en-v1.5, with masked language modeling pre-training, followed by weakly-supervised contrastive learning on large-scale web data. The contrastive fine-tuning stage incorporates supervision from diverse high-quality English datasets. Reversed RoPE scaling is employed during pre-training to handle longer sequences more efficiently.

Evaluation

stella_en_1.5B_v5 achieves strong results on the SentEval and BEIR benchmarks, often outperforming larger models. The elastic embeddings provide significant flexibility - the 1024d and 512d options perform comparably to the full 8192d embeddings in most cases, while being much more efficient.

5.3.4 Nextcloud-AI/multilingual-e5-large-instruct

Model Overview

multilingual-e5-large-instruct is a multilingual extension of the E5 text embedding model [Wang et al., 2023]. It has 24 transformer layers and produces 1024-dimensional embeddings. The model is initialized from XLM-RoBERTa-large and trained on a mixture of multilingual datasets with a focus on 30 languages.

Training Methodology

The training of multilingual-e5-large-instruct involves two main stages: Contrastive pre-training on 1 billion weakly-supervised text pairs Fine-tuning on datasets from the E5-mistral paper [Wang et al., 2023] For the fine-tuning stage, each text pair is prepended with a one-sentence instruction describing the task (e.g. "Given a web search query, retrieve relevant passages that answer the query"). This instruction tuning setup allows the model to adapt its representations for different use cases.

Evaluation

multilingual-e5-large-instruct demonstrates strong performance on both monolingual and cross-lingual benchmarks like MKQA, MLDR, and BEIR. On several non-English datasets, it outperforms much larger models like mDPR and E5-mistral-7b. The instruction tuning proves highly effective in aligning the embeddings for specific tasks.

5.3.5 BAAI/bge-small-en-v1.5

Model Overview

The bge-small-en-v1.5 model is part of the BGE (BAAI General Embeddings) series developed by the Beijing Academy of Artificial Intelligence [Xiao et al., 2023]. It is a compact English-specific model with just 25M parameters, making it highly efficient for deployment in resource-constrained environments. The model architecture is based on RoBERTa [Liu et al., 2019] with optimizations like dynamic token pruning and embedding factorization to reduce computational costs.

Training Methodology

bge-small-en-v1.5 follows a two-stage training pipeline similar to other BGE models:

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

The pre-training stage leverages diverse data sources like Wikipedia, Reddit, and CommonCrawl to learn general-purpose text representations. The fine-tuning stage incorporates datasets spanning retrieval, classification, paraphrase detection, and semantic textual similarity tasks to instill task-specific knowledge.

Evaluation

Despite its small size, bge-small-en-v1.5 punches above its weight on several English benchmarks. On the SentEval suite, it outperforms the base-sized BERT and RoBERTa models on most tasks while being 4x more compact. On retrieval challenges like BEIR, it achieves competitive results, often surpassing larger models like MPNet and the original DPR. 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

Model Size and Efficiency

The six models cover a broad spectrum of sizes, from the 25M parameter bge-small-en-v1.5 to the 7B parameter e5-multilingual-7b-instruct-v2. The smaller models (bge-small-en-v1.5, stella_en_1.5B_v5) excel in scenarios with strict latency or memory constraints, while the larger models (e5-multilingual-7b-instruct-v2, gte-large-en-v1.5) provide maximum accuracy for offline processing or applications demanding the highest quality embeddings. The base-scale models (jina-embeddings-v3, multilingual-e5-large-instruct) strike a good balance for most use cases, delivering strong performance with reasonable computational requirements. Notably, bge-small-en-v1.5 achieves impressive results despite its tiny size, making it a top choice for efficiency-focused applications.

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

Table 5.3: Comparison of Embedding Models

Language Coverage

Three models (gte-large-en-v1.5, stella_en_1.5B_v5, bge-small-en-v1.5) are designed specifically for English, while the others offer multilingual support. jina-embeddings-v3 covers 100+ languages with a focus on 30 major ones, and the E5 models support 100 languages but may see degraded performance on low-resource languages. For English-only applications, the specialized models are recommended, with bge-small-en-v1.5 being the most efficient and gte-large-en-v1.5 providing the highest quality. jina-embeddings-v3 is a strong choice when both English performance and broad language coverage are desired.

Supported Features

All models handle long input sequences (2048 tokens), which is important for document-level tasks. jina-embeddings-v3 includes dedicated task-specific

adapters for fine-grained control, while the E5 models leverage instruction prompts for easy task adaptation. stella_en_1.5B_v5 and jina-embeddings-v3 support Matryoshka embeddings for flexibly adjusting embedding sizes, and bge-small-en-v1.5 employs embedding factorization for better efficiency-quality trade-offs at low dimensions.

Empirical Performance

On English benchmarks, gte-large-en-v1.5 and stella_en_1.5B_v5 generally lead the pack, followed closely by bge-small-en-v1.5 which punches far above its weight class. jina-embeddings-v3 also shows strong English results while supporting many more languages. For multilingual tasks, e5-multilingual-7b-instruct-v2 achieves the highest absolute scores, while jina-embeddings-v3 and multilingual-e5-large-instruct offer the best performance-efficiency trade-offs. On long-context understanding, the E5 models and gte-large-en-v1.5 are top choices, with bge-small-en-v1.5 and stella_en_1.5B_v5 being less suitable due to their more limited sequence lengths.

Conclusion

In this extended comparative study, we analyzed five state-of-the-art text embedding models: gte-large-en-v1.5, jina-embeddings-v3, stella_en_1.5B_v5, multilingual-e5-large-instruct, and bge-small-en-v1.5. Key insights:

Model size presents a key trade-off: larger models offer maximum quality but at steep computational costs, while smaller models prioritize efficiency, with bge-small-en-v1.5 showing impressive performance-size ratio. For English-only applications, the specialized gte-large-en-v1.5, stella_en_1.5B_v5 and bge-small-en-v1.5 are top picks, while jina-embeddings-v3 and the E5 models are best for multilingual use cases. Models with flexible embedding sizes (stella_en_1.5B_v5, jina-embeddings-v3, bge-small-en-v1.5) enable powerful yet efficient representations for real-world applications. Instruction prompts (E5 models) and dedicated task adapters (jina-embeddings-v3) offer convenient mechanisms for adapting embeddings to specific downstream tasks. Empirically, all models show strong results on a wide range of benchmarks, with e5-multilingual-7b-instruct-v2 leading in absolute scores, gte-large-en-v1.5 and stella_en_1.5B_v5 excelling at English understanding, and jina-embeddings-v3 providing an excellent balance for multilingual applications.

Selecting the right embedding model requires careful consideration of the target use case, available computational resources, and deployment constraints. For English-centric applications prioritizing efficiency, bge-small-en-v1.5 is a top choice, while gte-large-en-v1.5 and stella_en_1.5B_v5 deliver maximum quality. jina-embeddings-v3 and the E5 models are recommended for scenarios requiring broad language coverage and task flexibility. As text embedding techniques continue to advance at a rapid pace, this comparative analysis aims to provide a snapshot of the current state-of-the-art and offer practical insights for practitioners. By understanding the strengths and trade-offs of each model, researchers and developers can make informed decisions when building embedding-powered applications. Looking ahead, we expect to see further innovations in model efficiency, task adaptation, and cross-lingual transfer, unlocking even more powerful and versatile text representations.

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 chapter examines three distinct chunking methods for RAG systems, each with its unique characteristics and potential advantages.

5.4.1 Parsing Documents into Text Chunks (Nodes)

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.

Advantages and Limitations

Advantages:

1. Simple to implement and understand
2. Consistent chunk sizes facilitate uniform processing

Limitations:

1. Fixed chunk sizes may not align with natural breaks in the text
2. Larger chunks can introduce irrelevant information and increase computational costs

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, documents are parsed into three levels of chunks:

- 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.

Advantages and Limitations

Advantages:

1. Allows for fine-grained retrieval with the option to expand context
2. Adapts to different levels of specificity required by queries

Limitations:

1. More complex to implement and manage
2. Increased storage requirements due to redundancy in the hierarchy

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 will examine two window sizes: 3 and 6.

Window Sizes

- Window Size 3: For each sentence, the chunk includes the sentence itself, one preceding sentence, and one following sentence.
- Window Size 6: For each sentence, the chunk includes the sentence itself, two preceding sentences, and three following sentences.

Advantages and Limitations

Advantages:

1. Preserves semantic units (sentences) and their immediate context
2. Adapts to the natural structure of the text

Limitations:

1. Variable chunk sizes may complicate processing and indexing
2. Optimal window size may vary depending on the document type and content

5.4.4 Evaluation

Each of the three chunking methods presented in this chapter 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

Retrieval Method, fdsjnfkdsn, jfiewdf ,fjneirufj

Method	Parameters	Random Sampling		Over Sampling	Avg.		Complete Run		
		Acc	F1	Acc	Acc	F1	Acc	F1	Latency
Original	Chuck Size: 1024	0.719	0.391	**0.450**	**0.212**	0.255	**0.528**	0.353	0.353
	Chuck Size: 512	0.719	0.391	**0.450**	**0.212**	0.255	**0.528**	0.353	0.353
	Chuck Size: 256	0.719	0.505	**0.450**	**0.212**	0.255	**0.528**	0.353	0.353
small2big	Text Chunks: 8 * 128, 4 * 256, 2 * 512	0.505	0.391	**0.450**	**0.212**	0.255	**0.528**	0.353	0.353
Sliding Window	Window Size: 6	0.719	0.505	**0.450**	**0.212**	0.255	**0.528**	0.353	0.353
	Window Size: 3	0.719	0.505	**0.450**	**0.212**	0.255	**0.528**	0.353	0.353

Table 5.4: Evaluation Results for Different Embeddings Models through the Pipeline (just with the Gemma2 model)

the appendix B for further reference.

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, and also we re-rank the nodes based on their similarity score with re-ranker models (*ms-marco-MiniLM-L-6-v2* and *jina-reranker-v2-base-multilingual*) and not their original score.

Algorithm 3 Similarity Cutoff Postprocessor

```
1:  procedure PostprocessNodes(nodes, knowledge_graph, similarity_cutoff)
2:      new_nodes ← []
3:      node_texts ← [node.text for node in nodes]
4:      re_rank_nodes ← ReRank(knowledge_graph, node_texts)
5:      for each node in nodes do
6:          node.score ← get_node_score(node.text, re_rank_nodes)
7:          if node.score > similarity_cutoff then
8:              new_nodes.Append(node)
9:          end if
10:     end for
11:     return new_nodes
12: end procedure
```

5.6 Evaluation

5.7 Failure Analysis

6

Conclusions and Future Works

A	**B**
C	D
E	F
G	H

Table 6.1: Table example

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
```

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_spouse_00134* entry from the FactBench dataset. We report the best node found by through our pipeline for each chunking strategy.

B.1 Text Splitter - Chuck Size 512

B.2 Small2Big

B.3 Sliding Window - Window Size 3

The knowledge graph triple *correct_award_00000* from the FactBench dataset with triple "Henry Dunant award Nobel Peace Prize" is used as an example. The model used for this example 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.

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.

Table B.1: Sliding Window - Window Size 3

References

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

[2] 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.

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

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] Xuan Zhang and Wei Gao. *Reinforcement Retrieval Leveraging Fine-grained Feedback for Fact Checking News Claims with Black-Box LLM*. 2024. arXiv: 2404.17283 [cs.CL]. URL: https://arxiv.org/abs/2404.17283.

Acknowledgments

I really want to thank my professors for all their help with my thesis. You guys were awesome!

Prof Silvello, you've got such a sharp mind. The way you break down tricky stuff and give feedback really helped me up my game. You've definitely shaped how I tackle problems now.

And Prof Marchesin, you're just the best. Your friendly vibes made me feel so welcome. I loved how you got excited about new ideas and encouraged me to think outside the box. It made working on this thesis way more fun and interesting.

Honestly, I couldn't have done this without both of you. Your guidance meant the world to me. Thanks for everything!