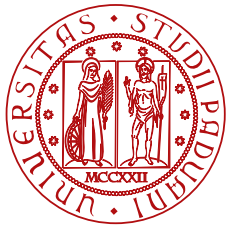


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

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

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

Abstract

Sommario

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

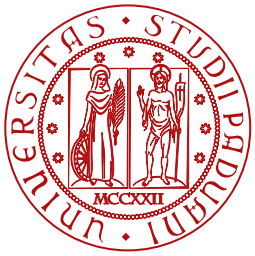
CSV Comma Separated Values

1

Introduction

2

State of the Art



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Figure 2.1: Example of image

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 natural language processing pipeline is the Knowledge Graph Dataset, a structure that serves as the primary source of information for subsequent processing stages. This section elucidates the nature, purpose, and significance of the knowledge graph within the context of our information retrieval and question-answering system.

3.1.1 Definition and Purpose

A knowledge graph is a structured representation of information that models real-world entities and their interrelations, organized in the form of a graph. In our pipeline, the Knowledge Graph Dataset functions as a comprehensive, interconnected repository of facts, concepts, and relationships. Its primary purpose is to provide a rich, contextual foundation for understanding and processing complex queries.

The utilization of a knowledge graph in our system serves several critical purposes:

- **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

Following the foundational Knowledge Graph Dataset, the Query Generation and Processing stage represents a critical juncture in our pipeline, where user inputs are transformed into structured queries that can be effectively processed by subsequent components. This section elucidates the mechanisms and methodologies employed in this crucial phase of the information retrieval process.

3.2.1 Human-Understandable Text Generation

The pipeline initiates with the generation of human-understandable text, a process that bridges the gap between raw data representation and natural language understanding. Key aspects of this process include:

- **Contextual Awareness:** Incorporating relevant context from the Knowledge Graph to ensure generated text is pertinent and informative.
- **Adaptability:** Tailoring 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.

Take note that some knowledge graph datasets are human-readable, while others are not, for example in the FactBench dataset, can be read and understood by humans easily by concatenating the subject, predicate, and object of each triple. Add Table of examples Here.

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. Add Table of examples Here. Add the prompt template here.

3.2.3 Cross-Encoder for Query Relevance Scoring

The Cross-Encoder component plays a pivotal role in assessing the relevance of the generated questions. As depicted in the Figure 1, this module takes multiple inputs and produces relevance scores for each question.

Key features of the Cross-Encoder include:

- Input Processing
  - Source Sentence: The original input text.
  - Sentences to Compare: Likely the 10 generated questions.
  - 1st Question to 10th Question: Individual assessment of each generated question.

- Scoring Mechanism: The Cross-Encoder assigns numerical scores (e.g., 0.83 as shown in the image) 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.

Input Processing: Add Table of exmples Here.

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 may loop back to generate new questions or adjust the existing ones, maintaining the quality of queries entering subsequent stages.

3.3 Information Retrieval Mechanisms

The Information Retrieval Mechanisms form a crucial component of our pipeline, bridging the gap between query processing and content synthesis. This stage is responsible for acquiring relevant information from both internal and external sources, creating a comprehensive data pool for subsequent analysis and response generation. 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: KOSSHER $_{H}ERE$ Paramerters

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 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.

3.3.3 Data Pool Creation fetch data - newspaper 4k - how it works ???

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 Context Processing and Chunking

As illustrated in the pipeline diagram, the retrieved information undergoes further refinement through context processing and chunking. This stage involves:

- Context Definition: Establishing the boundaries and relationships of contextual units within the retrieved information.
- Chunking Strategies: Breaking down large texts into smaller, manageable chunks that can be more effectively processed by subsequent stages of the pipeline.

Embedding and Retrieval Tasks The Embedding and Retrieval Tasks represent a crucial stage in our pipeline, bridging the gap between raw textual data and machine-interpretable vector representations. This component plays a vital role in enhancing the efficiency and accuracy of information retrieval and subsequent processing stages. As illustrated in the pipeline diagram, this section focuses on embedding for retrieval tasks, particularly for smaller chunks of information. 3.5.1 Embedding Techniques for Smaller Chunks The pipeline employs advanced embedding techniques to transform textual data into dense vector representations, facilitating more efficient and semantically aware retrieval processes. Key aspects of this embedding process include:

Granularity: The focus on "Smaller Chunks" suggests a fine-grained approach to embedding, where text is broken down into manageable units. This granularity allows for more precise retrieval and relevance assessment. Dimensionality Reduction: Transforming high-dimensional textual data into lower-dimensional vector spaces while preserving semantic relationships. Contextual Embeddings: Utilizing state-of-the-art models capable of capturing contextual nuances, such as BERT, RoBERTa, or more recent transformer-based architectures. Domain Adaptation: Potentially fine-tuning embedding models on domain-specific corpora to enhance performance in specialized knowledge areas.

Implementation considerations:

Model Selection: Choosing appropriate pre-trained models or developing custom embedding solutions tailored to the specific needs of the pipeline. Computational Efficiency: Balancing the trade-off between embedding quality and processing speed, especially important for real-time applications. Updating Mechanisms: Implementing strategies to periodically update or refine embedding models to accommodate evolving language usage and domain knowledge.

3.5.2 Similarity Cutoff Strategy A key feature highlighted in the pipeline diagram is the "Similarity Cutoff Strategy," which plays a crucial role in filtering and prioritizing embedded information. 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. Dynamic Thresholding: Potentially implementing adaptive thresholding mechanisms that adjust based on the query complexity, result set size, or other contextual factors. Cluster Analysis: Possibly utilizing clustering techniques to group similar embeddings and identify representative samples or outliers.

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.

3.5.3 Integration with Retrieval Tasks The embedding process is tightly coupled with retrieval tasks, forming a cohesive system for efficient information access and utilization. Key aspects of this integration include:

Index Construction: Building and maintaining efficient index structures (e.g., inverted indices, approximate nearest neighbor indices) to facilitate rapid retrieval of relevant embeddings. Query Embedding: Transforming incoming queries into the same embedding space as the document chunks, enabling semantic similarity comparisons. Retrieval Mechanisms: Implementing efficient algorithms for retrieving the most similar embedded chunks given a query embedding, potentially utilizing techniques like approximate nearest neighbor search. Re-ranking: Possibly employing a two-stage retrieval process where an

initial broad retrieval is followed by more computationally intensive re-ranking of the top results.

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

Handling Out-of-Vocabulary Words: Developing strategies to manage words or phrases not seen during the embedding model's training. 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 pretrained embedding models and ensuring fair representation across diverse topics and perspectives.

3.5.5 Connection to Subsequent Stages The output of the Embedding and Retrieval Tasks directly influences the effectiveness of subsequent pipeline stages:

Synthesis Input: Providing a refined, relevance-filtered set of information chunks to the LLM-based synthesis stage. Context Enhancement: Offering a mechanism for rapid contextual lookup during the response generation process, allowing the system to access relevant information efficiently. Diversity in Response Generation: Enabling the retrieval of diverse yet relevant information, contributing to more comprehensive and nuanced responses. window

In conclusion, the Embedding and Retrieval Tasks component represents a sophisticated approach to transforming and organizing textual information for efficient retrieval and utilization. By leveraging advanced embedding techniques and implementing strategic similarity cutoff mechanisms, this stage ensures that subsequent components of the pipeline have access to highly relevant and semantically rich information. The emphasis on processing smaller chunks and employing similarity-based filtering demonstrates a commitment to precision and efficiency in information retrieval, ultimately contributing to the overall effectiveness and intelligence of the question-answering system.

3.4 Large Language Models (LLMs) in the Pipeline

FS

write about Large Language Models (LLMs) in the Pipeline

Certainly. I'll now detail the role of Large Language Models (LLMs) in your pipeline, maintaining an academic tone and focusing on the elements depicted in the image.

3.6 Large Language Models (LLMs) in the Pipeline

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

3.6.1 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.

3.6.2 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, including the Knowledge Graph Dataset and retrieved web content. Context Processing: Analyzing and interpreting the broader context of queries and retrieved information to generate more accurate and relevant

responses. Natural Language Generation: Producing human-like text that forms the basis of the system's responses to user queries. Reasoning and Inference: Drawing logical conclusions and making inferences based on the available information, potentially filling gaps in explicit knowledge.

3.6.3 Majority Voting System

A key feature of the LLM integration, as depicted in the diagram, is the implementation of a majority voting system for response generation. This approach offers several benefits:

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.

3.6.4 Integration with Other Pipeline Components

The LLMs interact closely with other components of the pipeline:

Input from Embedding and Retrieval: The LLMs receive contextualized, relevant information chunks from the embedding and retrieval stages, ensuring they have access to the most pertinent data for response generation. Feedback to Information Retrieval: The LLMs may potentially provide feedback to refine or guide further information retrieval based on initial processing results. Interface with Final Judge: As indicated in the diagram, the output from the LLM majority voting system feeds into a "final judge" component, suggesting an additional layer of quality control or decision-making.

3.6.5 Challenges and Considerations

Several challenges and ethical considerations are associated with the use of

LLMs in the pipeline:

Bias Mitigation: Addressing and mitigating potential biases inherent in the training data of LLMs. Explainability: Developing methods to provide transparency and explanations for the reasoning behind generated responses, especially crucial in a multi-model voting system. Consistency: Ensuring consistency in responses across multiple queries and different combinations of activated LLMs. Computational Resources: Managing the significant computational requirements of running multiple LLMs concurrently. Privacy and Data Handling: Ensuring that sensitive information is handled appropriately and that the LLMs do not inadvertently reveal private data. Ethical Use: Implementing safeguards to prevent the generation of harmful, false, or misleading content.

3.6.6 Future Directions

Future work

The integration of LLMs in the pipeline opens up several avenues for future enhancements:

Dynamic Model Selection: Implementing systems that can dynamically select the most appropriate combination of LLMs based on the specific query and context. Continuous Learning: Exploring methods for ongoing fine-tuning or adaptation of the LLMs based on user interactions and feedback, while maintaining ethical boundaries. Multi-modal Integration: Expanding the capabilities of the pipeline to incorporate LLMs that can process and generate multi-modal content (text, images, audio). Enhanced Reasoning Capabilities: Developing techniques to improve the logical reasoning and inference capabilities of the LLMs, potentially through integration with symbolic AI approaches.

In conclusion, the incorporation of multiple Large Language Models into the pipeline represents a sophisticated approach to natural language processing and question answering. By leveraging the strengths of diverse models and implementing a majority voting system, the pipeline aims to generate more accurate, nuanced, and reliable responses. This approach, while powerful, also necessitates careful consideration of ethical implications and ongoing efforts to address challenges related to bias, consistency, and explainability. The central role of LLMs in this pipeline underscores their transformative potential in advancing the field of artificial intelligence and natural language understanding.

3.5 Model Diversity and Conflict Resolution

The integration of diverse models and the implementation of conflict resolution mechanisms are crucial components of our advanced natural language processing pipeline. This section explores the strategies employed to leverage model diversity and resolve conflicts, ensuring robust and reliable outputs. 3.7.1 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.

3.6 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. 3.8.1 Overall Pipeline Flow 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. 3.8.2 Input Processing and Query Generation The pipeline initiates with the processing of user input and the generation of structured queries. Key components and decision points: a) Knowledge Graph Dataset Integration:

The initial input is contextualized using the Knowledge Graph Dataset. Decision Point: Determining the extent and nature of knowledge graph integration based on the input complexity.

b) Human-Understandable Text Generation:

Transformation of structured data into natural language representations. Decision Point: Selecting the most appropriate natural language generation technique based on the input and context.

c) Question Generation:

The system generates 10 questions about the input sentence. Decision Point: Assessing the quality and relevance of generated questions.

d) Cross-Encoder for Query Relevance:

Scoring the relevance of generated questions. Critical Decision Point: Determining if the top question score exceeds the upper threshold.

If yes: Proceed to information retrieval. If no: Potential loop back for query refinement or regeneration.

3.8.3 Information Retrieval and Enrichment Upon successful query generation, the pipeline moves to retrieve and enrich relevant information. Key components and decision points: a) Google Search Integration:

Submission of top N questions and main question to Google Search. Retrieval of top 100 search results. Decision Point: Balancing between the breadth of search (number of questions submitted) and depth (number of results retrieved).

b) Process and Extract Links:

Extraction and processing of information from retrieved search results. Decision Point: Determining the relevance and quality of extracted information for inclusion in the data pool.

c) Data Pool Creation:

Aggregation of retrieved and processed information. Decision Point: Structuring the data pool for optimal accessibility in subsequent stages.

3.8.4 Embedding and Relevance Assessment This stage involves the transformation of textual data into vector representations and the assessment of relevance. Key components and decision points: a) Embedding for Retrieval Tasks:

Generation of embeddings for smaller chunks of information. Decision Point: Selecting the most appropriate embedding technique based on the nature of the data.

b) Similarity Cutoff Strategy:

Implementation of relevance thresholds for embedded information. Critical Decision Point: Determining the similarity cutoff threshold.

Information above the threshold: Retained for further processing. Information below the threshold: Potentially discarded or deprioritized.

c) Context Processing:

Chunking of context into manageable units. Decision Point: Determining the optimal chunk size and processing method.

3.8.5 Multi-Model Processing and Synthesis This stage leverages multiple Large Language Models (LLMs) for information processing and response generation. Key components and decision points: a) Parallel LLM Processing:

Simultaneous processing of information by multiple LLMs. Decision Point: Allocating specific tasks or aspects of the query to different models based on their strengths.

b) Synthesis of Information:

Integration of outputs from multiple LLMs. Decision Point: Determining the method of synthesis (e.g., concatenation, abstraction, or hybrid approaches).

c) Majority Voting System:

Aggregation of model outputs to determine the most appropriate response. Critical Decision Point: Assessing the level of agreement among models.

Clear majority: Proceed with the majority response. Lack of consensus: Trigger conflict resolution mechanisms.

3.8.6 Conflict Resolution and Final Output Generation The final stage of the pipeline focuses on resolving conflicts and generating the final response. Key components and decision points: a) Conflict Identification:

Detection of disagreements or inconsistencies in model outputs. Decision Point: Determining the threshold for what constitutes a significant conflict requiring resolution.

b) Adaptive Model Selection:

Critical Decision Point: When two models believe the answer is correct and two believe it's wrong, switching to a more advanced model.

Decision on which advanced model to use (commercial or larger parameter model).

c) Final Judge Implementation:

Resolution of remaining conflicts and ensuring output coherence. Decision Point: Determining the final response based on aggregated model outputs and conflict resolution results.

d) Response Generation:

Formulation of the final output to be presented to the user. Decision Point: Selecting the most appropriate format and level of detail for the response.

3.8.7 Feedback Loops and Iterative Processing Throughout the pipeline, several feedback loops and iterative processes are evident:

Query Refinement Loop: In cases where initial queries don't meet relevance thresholds, the system may loop back to generate new or refined questions. Information Retrieval Iteration: The system may perform multiple rounds of information retrieval and processing to gather sufficient relevant data. Model Consensus Feedback: The degree of consensus among models may trigger additional processing or the invocation of more advanced models.

3.8.8 Challenges and Considerations 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.

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

3.7 Performance Metrics and Evaluation

FS

write about Performance Metrics and Evaluation

Certainly. I'll now detail the Performance Metrics and Evaluation aspects of your pipeline, maintaining an academic tone and focusing on the elements depicted in the image, while also expanding on general best practices in AI system evaluation.

3.9 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.

3.9.1 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.

3.9.2 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.

3.9.3 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.

3.9.4 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.

3.9.5 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.

3.9.6 Ethical and Bias Evaluation

Given the ethical considerations in AI systems, specific metrics for assessing fairness and bias are crucial:

Demographic Parity: Ensuring consistent performance across different demographic groups. Bias Detection: Implementing tools to identify and quantify potential biases in responses. Toxicity Metrics: Evaluating the system's ability to avoid generating harmful or inappropriate content.

3.9.7 User Satisfaction and Real-world Performance

While not directly observable from the pipeline diagram, user-centric evaluation is essential:

User Feedback Mechanisms: Implementing systems to collect and analyze user satisfaction data. Task Completion Rate: Assessing how often users successfully accomplish their intended tasks using the system's responses. A/B Testing: Comparing different versions of the pipeline or specific components in real-world scenarios.

3.9.8 Continuous Evaluation and Monitoring

The complexity of the pipeline necessitates ongoing evaluation:

Performance Drift Detection: Implementing systems to detect degradation in performance over time. Automated Regression Testing: Regularly running a suite of test cases to ensure consistent performance across pipeline updates. Anomaly Detection: Developing mechanisms to flag unusual patterns in system behavior or output.

3.9.9 Comparative Evaluation

Benchmarking the pipeline against:

Previous Versions: Tracking improvements over time. Alternative Architectures: Comparing performance with different pipeline configurations. Industry

Standards: Evaluating against established benchmarks in the field of question-answering systems.

3.9.10 Challenges and Considerations

Several challenges are associated with comprehensive evaluation of such a complex system:

Metric Interdependence: Balancing potentially conflicting metrics (e.g., accuracy vs. latency). Evaluation Bias: Ensuring that the evaluation process itself doesn't introduce biases. Evolving Standards: Keeping evaluation metrics aligned with rapidly advancing field standards. Interpretability: Developing methods to explain and interpret complex performance metrics to stakeholders. Holistic Assessment: Integrating various metrics into a meaningful overall performance indicator.

In conclusion, the Performance Metrics and Evaluation of this advanced NLP pipeline involve a multifaceted approach, combining binary success criteria (PASSED/FAILED) with a range of nuanced metrics assessing accuracy, efficiency, consistency, robustness, and ethical considerations. The complexity of the pipeline, with its multiple models and decision points, necessitates a comprehensive and ongoing evaluation strategy. This approach not only helps in assessing the current performance but also guides continuous improvement efforts, ensuring that the system remains effective, efficient, and aligned with ethical standards in the dynamic field of AI and natural language processing. The explicit inclusion of PASSED and FAILED states in the pipeline diagram underscores the importance of clear, actionable performance assessments in the development and refinement of advanced AI systems.

3.8 Ethical Considerations and Limitations

The development and deployment of advanced natural language processing pipelines, such as the one described in this thesis, necessitate a thorough examination of ethical considerations and an acknowledgment of system limitations. This section explores the ethical implications of our question-answering system and discusses its inherent constraints. 3.10.1 Ethical Considerations While the pipeline diagram does not explicitly highlight ethical components, several aspects of the system raise important ethical considerations:

Bias in Knowledge Representation and Model Outputs

The pipeline's reliance on a Knowledge Graph Dataset and multiple Large Language Models (LLMs) introduces potential sources of bias: a) Knowledge Graph Biases:

Potential under- or over-representation of certain topics, cultures, or perspectives in the knowledge base. Risk of perpetuating historical biases present in the source data.

b) Model Biases:

LLMs may reflect biases present in their training data, potentially leading to skewed or unfair outputs. The use of multiple models, while potentially mitigating individual biases, may also amplify shared biases.

Mitigation Strategies:

Regular audits of the Knowledge Graph Dataset for representational fairness. Implementation of bias detection and mitigation techniques in model outputs. Diverse representation in the team developing and maintaining the system.

Privacy and Data Handling

The pipeline's information retrieval mechanisms, particularly the integration with Google Search, raise privacy concerns: a) User Query Privacy:

Ensuring that user queries are not stored or used in ways that could compromise individual privacy.

b) Data Retention Policies:

Establishing clear guidelines for how long retrieved information is stored and how it is used.

c) Handling of Sensitive Information:

Implementing safeguards to prevent the system from exposing or mishandling sensitive personal information that may be encountered during information retrieval.

Mitigation Strategies:

Implementation of robust data anonymization techniques. Clear and transparent data handling policies. Regular security audits and compliance checks with data protection regulations.

Transparency and Explainability

The complexity of the pipeline, particularly the use of multiple LLMs and a "final judge" for conflict resolution, presents challenges in terms of transparency: a) Black Box Decision-Making:

Difficulty in explaining how specific responses are generated, especially when advanced models are invoked for conflict resolution.

b) Accountability:

Challenges in attributing responsibility for erroneous or biased outputs in a multi-model system.

Mitigation Strategies:

Development of interpretable AI techniques to provide insights into decision-making processes. Implementation of logging systems to track the contribution of different components to the final output. Clear communication to users about the system's capabilities and limitations.

Misinformation and Harmful Content

The system's ability to synthesize information from various sources poses risks related to misinformation: a) Propagation of False Information:

Potential for the system to inadvertently spread misinformation present in retrieved data.

b) Generation of Harmful Content:

Risk of producing responses that could be considered harmful, offensive, or inappropriate.

Mitigation Strategies:

Implementation of fact-checking mechanisms and reliable source prioritization. Content filtering systems to detect and prevent the generation of harmful or inappropriate responses. Regular updates to the system to address emerging misinformation trends.

Environmental Considerations

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

High energy usage associated with running complex AI models and large-scale data processing.

b) Carbon Footprint:

Environmental impact of the infrastructure required to support the pipeline.

Mitigation Strategies:

Optimization of model efficiency and resource utilization. Exploration of more energy-efficient hardware and green computing practices. Consideration of the trade-offs between model complexity and environmental impact.

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

Scope of Knowledge

a) Temporal Limitations:

The Knowledge Graph Dataset and pre-trained models have a cutoff date for their information. Challenges in providing up-to-date information on rapidly evolving topics.

b) Domain Specificity:

Potential gaps in specialized or niche areas of knowledge. Variability in performance across different subject domains.

Language and Cultural Limitations

a) Language Coverage:

Potential biases towards languages well-represented in training data. Challenges in handling nuances of less common languages or dialects.

b) Cultural Context:

Limitations in understanding and appropriately responding to culturally specific queries or contexts.

Reasoning and Inference Capabilities

a) Complex Reasoning:

Limitations in handling queries requiring advanced logical reasoning or domain-specific expertise.

b) Causal Understanding:

Challenges in inferring causal relationships beyond correlations present in the training data.

Handling of Ambiguity and Context

a) Contextual Nuances:

Difficulties in capturing subtle contextual cues that humans naturally understand.

b) Disambiguation:

Challenges in resolving ambiguities in queries without additional user input.

Real-time Adaptation

a) Static Knowledge Base:

Limitations in adapting to real-time changes in the world without system updates.

b) Learning from Interactions:

The system's inability to learn and improve from individual user interactions due to privacy and architectural constraints.

Emotional and Social Intelligence

a) Empathy and Emotional Understanding:

Limitations in recognizing and appropriately responding to emotional cues in user queries.

b) Social Norms:

Challenges in adhering to complex and evolving social norms across different cultures and contexts.

3.10.3 Addressing Ethical Concerns and Limitations To address these ethical considerations and limitations, several approaches can be implemented:

Ethical Oversight:

Establishment of an ethics board to guide development and deployment decisions. Regular ethical audits of the system's performance and impacts.

User Education:

Clear communication to users about the system's capabilities, limitations, and potential biases. Promotion of critical thinking and fact-checking when using AI-generated information.

Continuous Improvement:

Ongoing research and development to address identified limitations and ethical concerns. Regular updates to the Knowledge Graph Dataset and model fine-tuning to improve accuracy and reduce biases.

Collaborative Development:

Engagement with diverse stakeholders, including ethicists, domain experts, and potential user groups, in the ongoing development and refinement of the system.

Regulatory Compliance:

Proactive alignment with emerging AI regulations and ethical guidelines. Participation in industry initiatives for responsible AI development.

In conclusion, while the advanced natural language processing pipeline presented in this thesis offers powerful capabilities for information retrieval and question answering, it also comes with significant ethical considerations and inherent limitations. Acknowledging these challenges is crucial for responsible development and deployment. By actively addressing ethical concerns, transparently communicating limitations, and continuously striving for improvement, we can work towards a system that not only advances the field of AI but also aligns with societal values and ethical standards. The complexity of these considerations underscores the need for ongoing dialogue and collaboration between technologists, ethicists, policymakers, and the broader public to ensure that AI systems like this pipeline contribute positively to society while

minimizing potential harms.

4 Analysis

4.1 A section

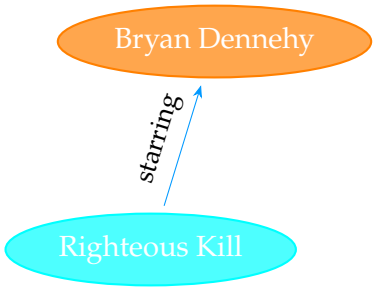


Figure 4.1: Image created with TikZ

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

graphicx

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 Document Selection

We explore various techniques for retrieving relevant documents from search engine results, with a specific focus on Google search engine. The goal is to identify the most effective methods for finding documents that perfectly match the information need expressed in the query. We consider both unsupervised and supervised approaches.

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.[1] 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 1 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 [2] spanning 26 languages, while also exhibiting superior NDCG@10 scores on English-language tasks in the BEIR benchmark [4]. 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 [3], 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.

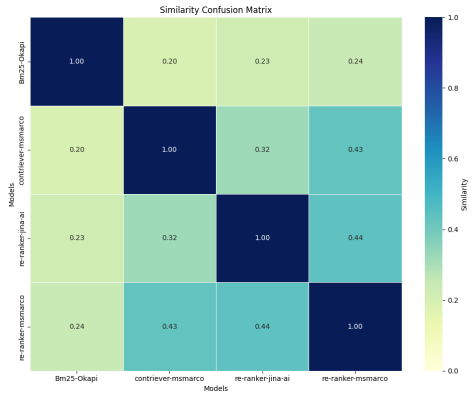


Figure 5.1: Document Retrieval Confusion Matrix

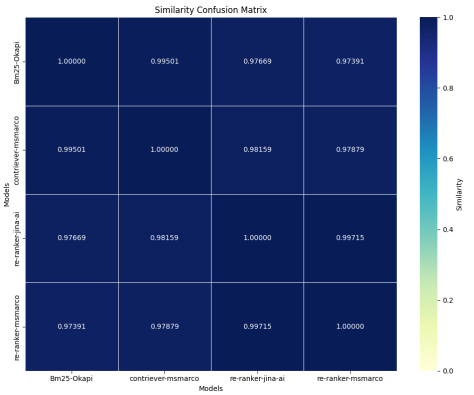


Figure 5.2: Document Retrieval Performance

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!** (**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 A 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 2 Similarity Cutoff Postprocessor

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

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

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.

A.1 Text Splitter - Chuck Size 512

A.2 Small2Big

A.3 Sliding Window - Window Size 3

Window - Highlighted Text is Original Text
Born: 15 September 1985, Cairo, Egypt Nationality: Egyptian Spouse: Hady El Bagory Movies and TV shows: Balash Tbousny (2017), Factory Girl (2013), Born to a Man 2016), Hepta: The Last Lecture (2016), Wahed Saheh (2011) 77. Yasmin Abdel Aziz is an Egyptian actress. Born: 16 January 1980, Cairo, Egypt Height: 1.73 m Spouse: Ahmed El-Awady (m. 2020), Mohamed Nabil Halawa (m. 2001–2018) Children: Yasmin Halawa, Seif El-Deen Halawa TV shows: Emraa Men Zaman El-Hob, Fawazeer El-Eyal Etganenet, Al-Raqs ala Slalem Mothareka, Ott Wa Far Movies: El-Anesah Mami (2012), Zaky Chan (2005), Aldada Dodi (2008), Karim's Harem (2005), The Hostage (2006) 78. Yosra El Lozy, is an Egyptian actress.She has received many awards from regional and international film festivals.
Angham Mohamed Ali Suleiman, known by the mononym Angham, is an Egyptian singer, recording artist, and actress. Her debut was in 1987 under the guidance of her father, Mohammad Suleiman. Born: 19 January 1972, Alexandria, Egypt Spouse: Ahmed Ezz (m. 2011–2012), Fahd Mohamed Al-Shalabi (m. 2004–2008), Magdy Aref (m. 1999–2000) Children: Abdel-Rahman Fahd Mohamed Al-Shalabi, Omar Aref Siblings: Ghenwa Mohammad Ali Suleiman, Khaled Mohammad Ali Suleiman, Ahmad Mohammad Ali Suleiman 4. Wartanoush Garbis Selim, better known by her stage name Anoushka, is an Egyptian singer and actress.
He explained The Palestinian situation is tough, Gaza, Jerusalem and West Bank are sieged. We suffer from what happens in Gaza and the division heavy burden for the last 10 years. In order to get rid of this suffering and siege, the leadership presented Hamas three points to response to: dissolve shadow government, form national unity government with Hamas as part and run legislative and presidential elections. We are ready for reconciliation tomorrow if the points were approved, but we haven't get any response so far. Abu Mazen is closer than Tehran, Europe and regional alliances. Why do we ask for others help while we're ready to reconciliation?

Table A.1: Evaluation Results for Different Methods through the Pipeline (just with the Gemma2 model)

References

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

[2] 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.

[3] 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.

[4] 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.

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!